FT62F08X

Application note

目录

FT62F08X SPI 应用

1. SPI 接口

1.1. 功能特性

- 3 线全双工同步传输
- 2 线半双工同步传输，或单向传输
- 主机模式或从机模式操作
- nss pin 软件或硬件管理
- 可编程的同步时钟极性和相位控制
- 可编程的 LSB first 或 MSB first
- 配置模式错误和 overrun 标志
- 硬件 CRC 校验支持
- Wakeup 唤醒支持

1.2. 功能描述

1.2.1. 一般描述

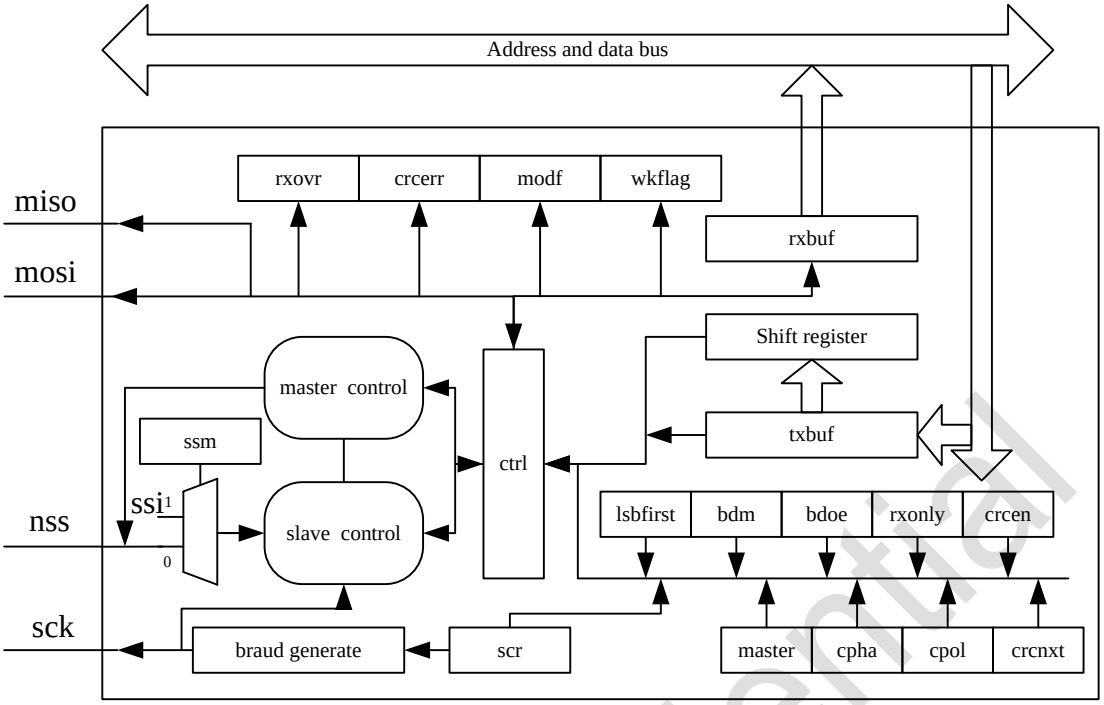

图 13.1 SPI 结构框图

SPI 接口总共有四个引脚：

- NSS:从机选择引脚低电平有效，在 NSS 引脚采用硬件管理用作输入时，如果 NSS 引脚为零则表示选择该从机，如果该接口配置为主机模式，则会产生 MODF 标志位，表示配置错误，这时该接口自动进入从机模式，这种特性用于兼容多主机通信；
- MOSI:主机数据输出/从机数据输入；
- MISO:主机数据输出/从机数据输入；
- SCK:主机串行时钟输出/从机串行时钟输入；

SPI 接口引脚通常的接法是采用三线全双工和四线全双工，如下图所示的引脚接法，NSS 引脚可以根据 NSSM 的值选择作为输入，或者是输出；当 NSS 引脚用作输入时，输入的值取决于 SSM 的值，当 SSM 置 1 时，NSS 引脚的值无效，SSI 的值会被传输到通信控制单元。SPI 模块支持双线半双工模式，主机模式时采用的数据通信引脚是 MOSI，从机模式时采用的数据通信引脚是 MISO。

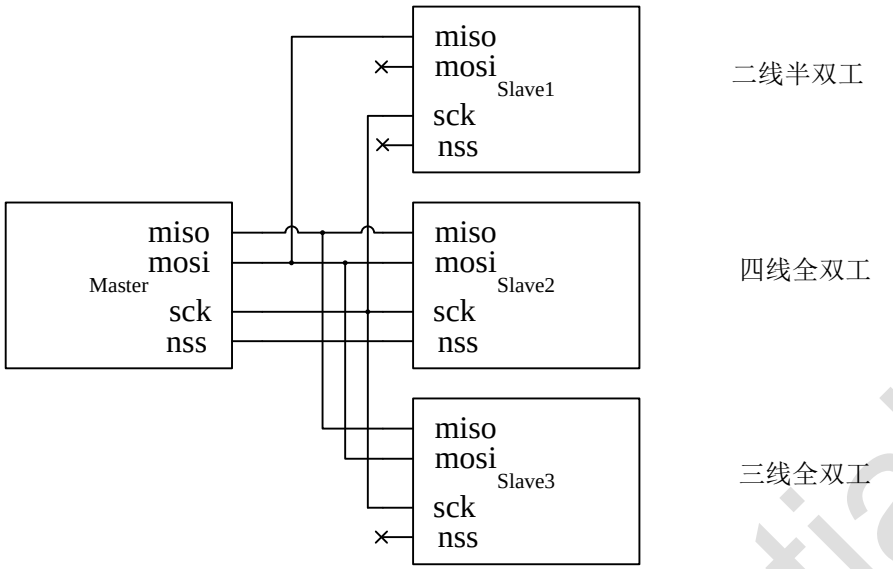

图 13.2 SPI 模块连接示意图

SPI 通信的时钟相位和极性设置如下图所示，时钟和相位的控制总共有四种情况，可以通过 CPHA 和 CPOL 设定相应配置，其中 CPOL 是控制模块空闲时的 SCK 的电平，当 CPOL 为 1 时，SCK 空闲时的电平为高电平，相反为低电平，图中所示箭头的位置是接收数据采样点；当使用了 NSS 引脚时，NSS 为低电平时才会接收数据；发送方串行数据的发送格式可以根据 LSBFIRST 来控制，默认情况下是先发送的八位数据的高位，最后发送的是八位数据的低位。

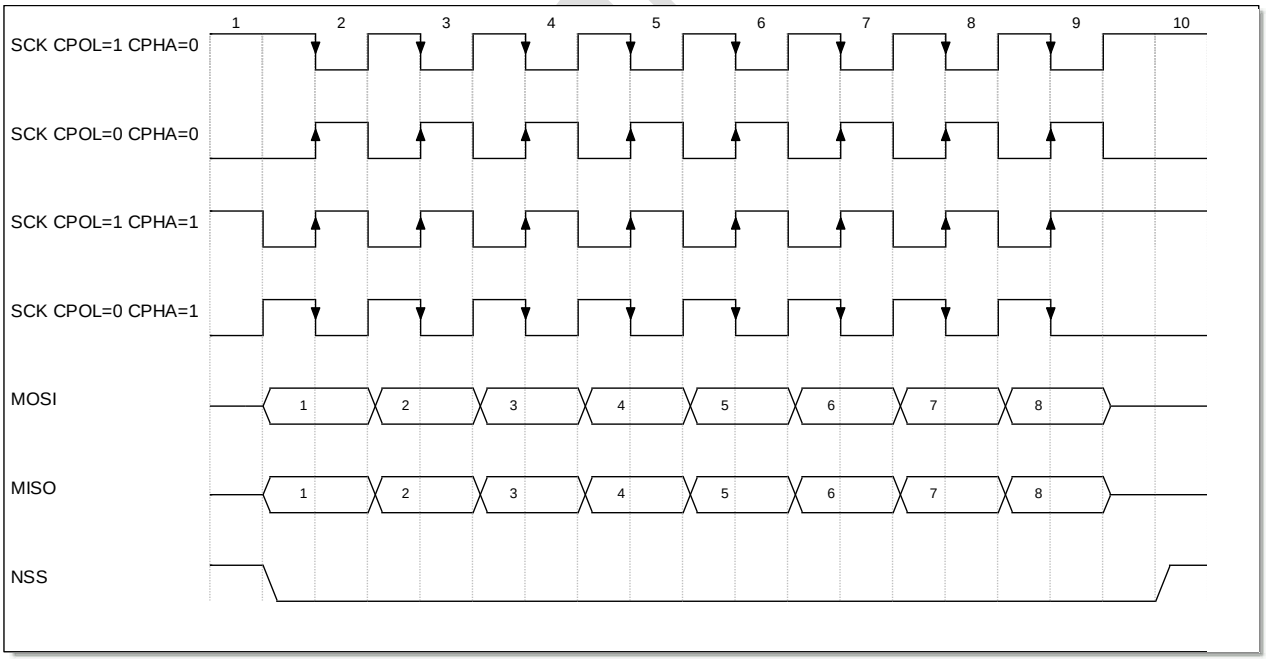

图 13.3 时钟极性和相位时序图

CRC 校验模块用来增强数据传输的可靠性，计算模块初始化的值采用的是零，默认多项式 CRCPOL 是 0x07，每次 CRCEN 从零到置位的时候都会对 CRC 模块进行初始化（该初始化不会影响 CRCPOL 的值），模块内部中间运算的值会丢失（RXCRC 和 TXCRC 的值会被置为零）。当 CRCEN 使能的时候，每次正常写入到 TXBUF 的值都会被送到 CRC 模块用来生成 TXCRC 的值，同样的在接收数据时每次正常写入到 RXBUF 的值也会被送到 CRC 模块，用来生产

RXCRC 的值；当需要传输 CRC 字节的时候，可以置位 CRCNXT，在正常数据传输完成后，下一次传输就会自动把 TXCRC 的值写入到 TXBUF(本次写入到 TXBUF 中的值不会送到 CRC 模块进行计算)，同时 CRCNXT 的值自动清零，在传输最后的 CRC 检验码的同时，也会接收对方的 CRC 检验码（本次接收到的数据不会写入 RXBUF），在接收完成时会比较 RXCRC 与接收到的校验码值，如果不匹配就会产生 CRCERR 标志位。

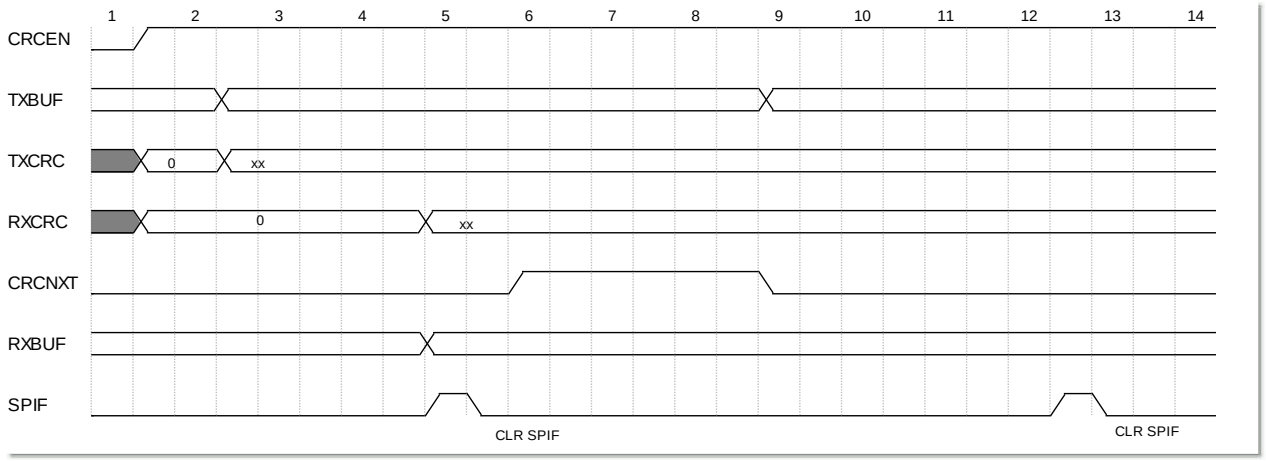

图 13.4 CRC 模块的工作时序图

1.2.2. 配置 SPI

主机模式的 SCK 是由主机产生的，从机的 SCK 是从机输入的，下面是 SPI 配置流程:

1. 根据主从机的需求配置 SPICFG 寄存器中的 MSTEN 位，置 0 是用作从机，置 1 是用作主机
2. 如果用作主机需要配置 SCR 寄存器，来配置通信的速率，通信的比特率 =Fmater/(2*(SCR+1)),
   该位对从机没有作用，只是在要求从机接收速率较快时，要提高 Fmaster 的频率，以便从机模块可以采样到 SCK 的上升沿或下降沿
3. 配置 NSSM 来设定如何使用 NSS 引脚，如果是用作四线主机模式则 NSS 引脚应配置为输出模式；如果用作从机模式，则可以配置成输入模式。如果只是用三线通信则可以设置禁用 NSS 引脚；另外 NSS 引脚在配置为输入模式时，可以置位 SSM 来启用软件管理 NSS 引脚的输入值，屏蔽掉实际的 NSS 引脚的值
4. 配置 SPICFG 中的 CPOL 和 CPHA 来配置 SCK 的相位和极性
5. 配置 SPICFG2 中的 LSBFIRST 来设置数据的传输格式
6. 配置 CRCPOL 寄存器和 CRCEN，使能 CRC 校验
7. 置位 SPICFG2 中的 RXONLY 只允许接收或者置位 BDM 来启用半双工通信
8. 置位 SPICFG 中的 SPIEN 来启用 SPI 通信接口，这时相应的 GPIO 接口会用作 SPI 通信接口，同时 SPIEN 从低电平到高电平的变化会导致清零 RXOVER，CRCERR，MODF，SPIF，WCF 标志位，置位 TXBMT，RXBMT 标志位
9. 如果使用中断模式进行通信则需配置 SPIIER 寄存器来使能相应的中断

1.2.3. 数据处理流程

数据通信的流程大致分为阻塞模式通信和非阻塞式模式通信，大致的处理方式是一样的，只是非阻
模式是在中断中进行的：

1. 阻塞通信时向 DATA 寄存器写入数据后需要查询 TXBMT，在查询到 TXBMT 为 1 时可以写入下一个数据；在 TXE 中断使能允许的时候，TXBMT 置 1 会直接进入中断
2. 阻塞模式接收数据时，需要一直查询 RXMBT，在查询到该位为 1 时，则可以读取 DATA 寄存器的值；在 RXNE 中断允许时，RXMBT 置 1 时则会进入中断
3. 阻塞模式接收的过程中需要查询 RXOVRN 和 CRCERR 位，在查询到相应的位置 1 后则需要写零清理相应的错误标志位；在 RXERR 中断使能时，发生相应的错误标志位会直接进入中断
4. 非阻塞模式中，进入中断后查询一次状态信息，然后根据相应的状态信息处理发送接收流程，处理完成以后退出中断子程序，继续处理其他的事情

无论是阻塞模式还是非阻塞模式，通信模块中的相关标识位变化如下图所示，在数据写入到发送数据寄存器后，TXBMT 从 1 变为 0，然后数据寄存器中的数据会传送到内部的移位寄存器，移位寄存器标志位从 1 变为 0，直到数据从移位寄存器中完全移出后变为 1；在发送的过程中，BUSY 状态位一直为 1；发送完成标志位在当前字节传输完成后会变为 1，同时 RXBMT 的值会从 1 变为 0。

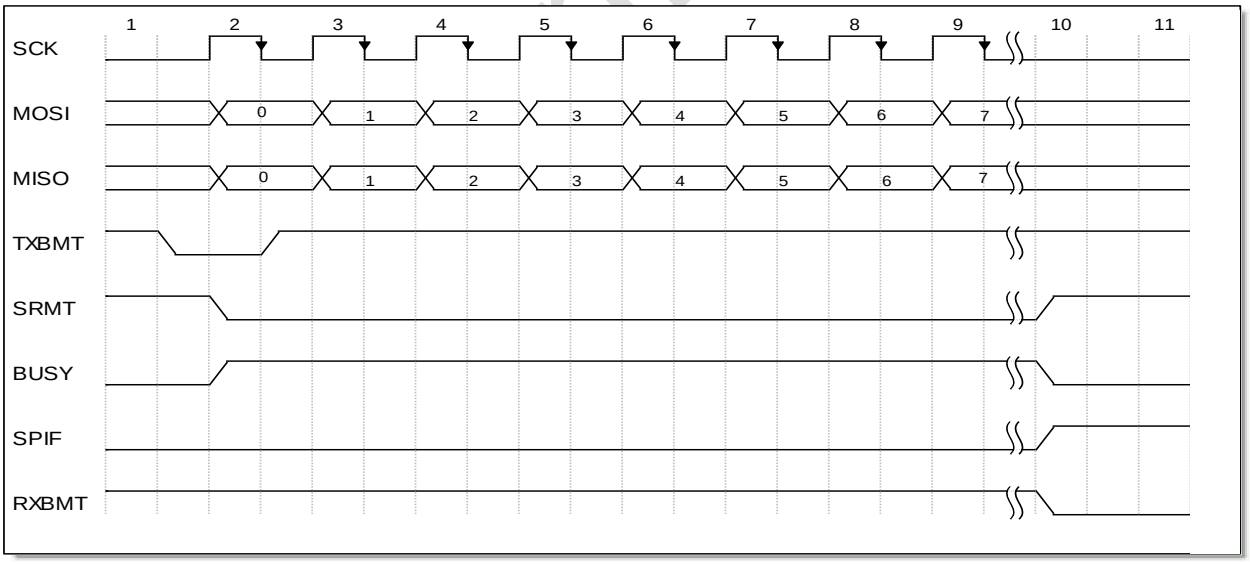

图 13.5 数据处理时序图

1.2.4. 睡眠模式唤醒

当系统进入睡眠模式，外设时钟存在时，SPI 模块有能力唤醒 MCU；如图所示，SPI 模块作为从机模块使用，开启了 WAKUP 中断使能，从机在接收到第一比特的数据时就会产生 WAKEUP 中断信号，在中断控制模块中如果使能了外设接口中断，就会直接唤醒 MCU；

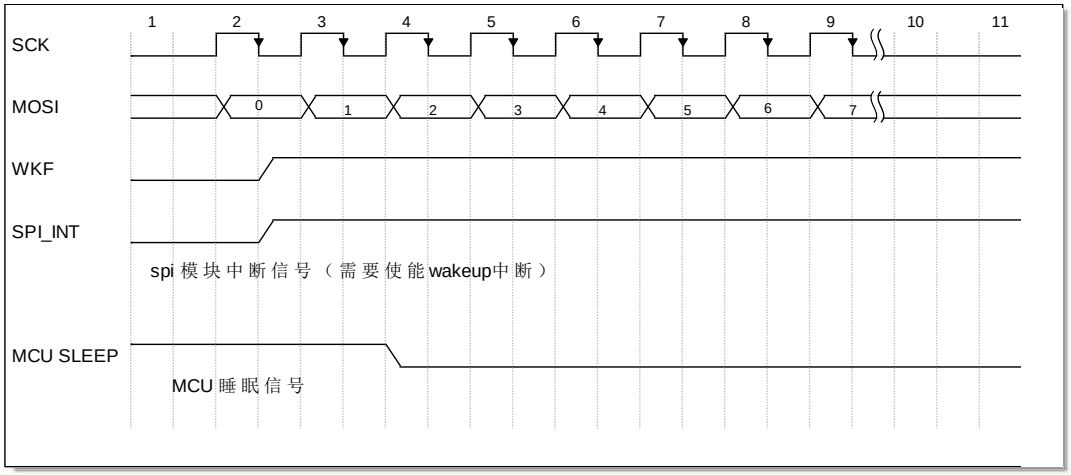

图 13.6 睡眠唤醒时序图

1.2.5. CRC 处理流程

阻塞模式传输数据时，在传输完成最后一个数据时，查询 TXBMT 的状态，在 TXBMT 位 1 时，则
置位 CRCNXT，这时 TXCRC 的值就会自动传输到 DATA 寄存器，然后 CRCNXT 就会自动清零， 查询 CRCNXT，查询到为 0 则清零 SPIF 状态位，然后再查询 SPIF 位，如果状态位置 1，则表示 CRC 校验码发送完成，接着查询 CRCERR 状态位，如果该状态位为 1，表示 CRC 校验码不匹配，写零清零相应的状态位；采用非阻塞模式通信时，TXBMT 为 1 进入中断时，若数据已经发送完整，则置位 CRCNXT，CRC 发送完成后如果产生了 CRCERR，则会直接进入中断，查询相关的标志位，然后写零清零。

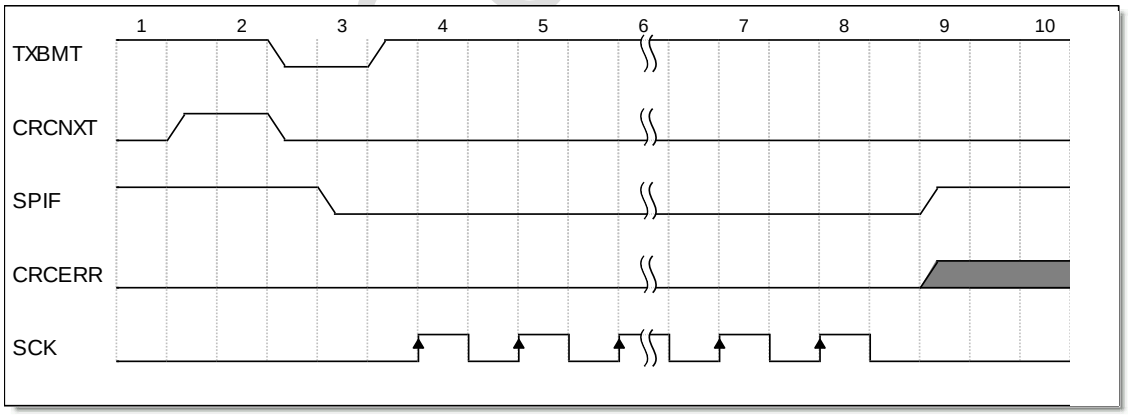

图 13.7 CRC 模块标志位时序图

1.3. 寄存器汇总

名称	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
SPIDATA	015h	DATA[7:0]								0000 0000
SPICTRL	016h	SPIF	WCOF	MODF	RXOVRN	NSSM		TXBMT	SPIEN	0000 0010
SPICFG	017h	BUSY	MSTEN	CPHA	CPOL	SLAS	NSSVAL	SRMT	RXBMT	0000 0000

SPISCR	018h	SCR[7:0]								0000 0000
SPICRCPOL	019h	CRCPOL[7:0]								0000 0111
SPIRXCRC	01Ah	RXCRC[7:0]								0000 0000
SPITXCRC	01Bh	TXCRC[7:0]								0000 0000
SPIIER	01Ch	—				WAKUP	RXERR	RXNE	TXE	---- 0000
SPICTRL2	01Dh	BDM	BDOE	RXONLY	SSI	SSM	CRCNXT	CRCEN	LSBFIRST	0000 0000
SPISTAT	01Eh	—	SMODF	SRXOVRN	SBUSY	SRXBMT	STXBMT	WKF	CRCERR	-000 0000

1.3.1. SPIDATA 寄存器，地址 0x015

Bit	7:0
Name	DATA
Reset	0x00
Type	RW

Bit	Name	Function
7:0	DATA	数据发送/接收寄存器（BUF）

1.3.2. SPICTRL 寄存器，地址 0x016

Bit	7	6	5	4	3:2	1	0
Name	SPIF	WCOL	MFAULT	RXOVRN	NSSM	TXBMT	SPIEN
Reset	0x0	0x0	0x0	0x0	0x01	0x1	0x0
Type	RW	RW	RW	RW	RW	RO	RW

Bit	Name	Function
7	SPIF	传输完成标志 0：表示没有传输完成或者已经清零 1：传输完成标志位，写零清理，写 1 无效
6	WCOL	BUF 写入失败标识， 0：BUF 写入正常 1：BUF 为非空时，进行写入会置位该位，写零清理，写 1 无效
5	MFAULT	工作模式错误标识， 0：工作模式正常 1：当 SPI 配置为主机模式，并且 NSS 用作输入引脚时，若 NSS 引进为低电平就会产生置位该位 写零清零，写 1 无效
4	RXOVRN	接收溢出标志 0：接收正常 1：接收溢出，写零清理，写 1 无效

3:2	NSSM	NSS 引脚模式选择，当用作输出模式时，相应的 IO 脚会被用作 NSS 输出 00:禁用 NSS 引脚 01:NSS 引脚用作输入 1x:NSS 引脚用作输出，输出的值等于 NSSM[0]的值 注：NSS 引脚配置为输入状态时，可以被 SSM 的软件管理模式屏蔽掉
1	TXBMT	发送 BUFF 为空状态 0：发送 BUF 非空 1：发送 BUF 位空
0	SPIEN	SPI 接口使能 0：禁用 SPI 模块 1：启用 SPI 模块，相应的 IO 会被用作 SPI 的功能 **注：在使用完 SPI 以后，禁用 SPI 不会对已经产生的标志位进行复位，只有再次打开 SPI 时会对标志位进行复位操作；SPIEN 从低电平到高电平的变化会导致复位标志位**

1.3.3. SPICFG 寄存器，地址 0x017

Bit	7	6	5	4	3	2	1	0
Name	BUSY	MSTEN	CPHA	CPOL	SLAS	NSSVAL	SRMT	RXBMT
Reset	0x00	0x0	0x0	0x0	0x0	0x1	0x1	0x1
Type	RO	RW	RW	RW	RO	RO	RO	RO

Bit	Name	Function
7	BUSY	SPI BUSY 状态 0：SPI 模块空闲 1：表示 SPI 模块忙绿中
6	MSTEN	MASTER 使能位 0：工作在 SLAVE 模式 1：工作在 MSATER 模式
5	CPHA	SCK 相位选择 0：第一个时钟转换的沿是数据采样点 1：第二个时钟转换的沿是数据采样点
4	CPOL	SCK 极性选择 0：SPI 空闲时，SCK 的时钟是处于低电平状态 1：SPI 空闲时，SCK 的时钟是处于高电平状态
3	SLAS	SLAVE 选择标志 0：该模块未被选中 1：该模块被选中 注：当 NSS 用作输入时，该值可以被 SSM 软件管理，当 SSM 为 1 时，这里的值表示的是 SSI 的值取反
2	NSSVAL	NSS 引脚的输入值状态 注：当 NSS 用作输入时，该值可以被 SSM 软件管理，当 SSM 为 1 时，这里的值表示的是 SSI 的值

1	SRMT	移位寄存器为空状态 0：内部串行移位寄存器非空 1：内部串行移位寄存器为空
0	RXBMT	接受 BUFFER 为空状态 0：表示接收 BUF 非空 1：接收 BUF 为空状态

1.3.4. SPISCR 寄存器，地址 0x018

Bit	7:0
Name	SCR
Reset	0x00
Type	RW

Bit	Name	Function
7:0	SCR	波特率设置寄存器,波特率=Fmaster/(2*(SCR+1)) 注：Fmaster 指的是外设时钟

1.3.5. SPICRCPOL 寄存器，地址 0x019

Bit	7:0
Name	CRCPOL
Reset	0x07
Type	RW

Bit	Name	Function
7:0	CRCPOL	CRC 计算多项式,默认值为 0x07

1.3.6. SPIRXCRC 寄存器，地址 0x01A

Bit	7:0
Name	RXCRC
Reset	0x00
Type	RO

Bit	Name	Function
7:0	RXCRC	接收数据的 CRC 计算结果 注：该寄存器在 CRCEN 发生从零到 1 的变化时会清零

1.3.7. SPITXCRC 寄存器，地址 0x01B

Bit	7:0
Name	TXCRC
Reset	0x00
Type	RO

Bit	Name	Function
7:0	TXCRC	发送数据的 CRC 计算结果 注：该寄存器在 CRCEN 发生从零到一的变化时会清零

1.3.8. SPIIER 寄存器，地址 0x01C

Bit	7:4	3	2	1	0
Name	—	WAKUP	RXERR	RXNE	TXE
Reset	—	0x0	0x0	0x0	0x0
Type	RO-0	RW	RW	RW	RW

Bit	Name	Function
7:4	N/A	保留位，读 0
3	WAKUP	唤醒中断使能 0：禁用唤醒中断 1：启用中断唤醒
2	RXERR	接收错误中断使能，包括 CRC 错误,接收溢出错误，模式错误 0：禁用接收数据出现 CRC 错误，溢出错误和模式错误中断 1：允许接收数据出现 CRC 错误，溢出错误和模式错误中断
1	RXNE	接收 BUF 不为空中断使能 0：禁用接收 BUF 不为空中断 1：允许接收 BUF 不为空中断
0	TXE	发送 BUF 为空中断使 0：禁用发送 BUF 为空中断 1：使能发送 BUF 为空中断

1.3.9. SPICTRL2 寄存器，地址 0x01D

Bit	7	6	5	4	3	2	1	0
Name	BDM	BDOE	RXONLY	SSI	SSM	CRCNXT	CRCEN	LSBFIRST
Reset	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Type	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	Function
7	BDM	半双工使能 0：禁用半双工模式 1：使能半双工模式
6	BDOE	半双工模式的接收使能 0：半双工模式接收使能 1：半双工模式发送使能
5	RXONLY	全双工模式只允许接收使能 0：全双工模式允许发送和接收 1：全双工模式只允许接收
4	SSI	NSS 输入管脚的值，仅当 SSM 置 1 时有效 0：输入到 NSS 引脚的值是 0 1：输入发哦 NSS 引脚的值是 1
3	SSM	软件 SLAVE 模式管理，使能后 NSS 引脚的值由 SSI 替代 0：禁用 NSS 引脚的软件管理模式 1：启用 NSS 引脚的关键管理模式，如果 NSS 引脚用作输入，则 NSS 引脚实际的值有 SSI 取代
2	CECNXT	置位后在 TXBUFF 为空时会把 TXCRC 的值写入 TXBUFF 0：不传送 TXCRC 的值到 TXBUF 1：等待 TXBMT 为 1 时，传送 TXCRC 的值到 TXBUF，写入完成后改位自动清零
1	CRCEN	CRC 模块计算使能 0：禁用 CRC 校验模块 1：启用 CRC 校验模块
0	LSBFIRST	发送低比特位使能 0：高比特位优先发送 1：低比特位优先发送

1.3.10. SPISTAT 寄存器，地址 0x01E

Bit	7	6	5	4	3	2	1	0
Name	—	SMODF	SRXOVRN	SBUSY	SRXBMT	STXBMT	WKF	CRCERR
Reset	—	0x0	0x0	0x0	0x1	0x1	0x0	0x0
Type	RO-0	RO	RO	RO	RO	RO	RW	RW

Bit	Name	Function
7	N/A	保留位，读 0
6	SMODF	同 SPICTRL[5]
5	SRXOVRN	同 SPICTRL[4]
4	SBUSY	同 SPICFG[7]
3	SRXBMT	同 SPICFG[0]
2	STXBMT	同 CTRL[1]

1	WKF	睡眠模式下，从机在接收数据时，会产生 WAKEUP 唤醒标志，写 0 清零，写 1 无效 0：没有发生 WAKEUP 唤醒或者已被清零 1：发生了 WAKEUP 唤醒事件
0	CRCERR	CRC 错误标志，写 0 清零，写 1 无效 0：传输过程中没有发生 CRC 校验错误或者已被清零 1：传输过程中发生了 CRC 校验错误

2 应用范例

```
//*************************************************************
/*   文件名: TEST_62F08x_SPI.c
*    功能：    FT62F08x-SPI 功能演示
*    IC:      FT62F088 LQFP32
*    内部：    16M
*    empno: 500
*    说明：    此演示程序为 60F12x_SPI 的演示程序.
*              该程序写 0x55,0x88 到(FT25C64A)0x13,0x14 地址,然后读取 0x13,0x14 的地址值
*
*
*    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     CS RB1

//volatile    unchar mydata; //全局查看变量定义
volatile unchar       SPIReadData=0;
/*-----------------------------------------------
 *   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 = 0B11111110;      //PA 输入输出  0-输出  1-输入
    PORTB = 0B00000000;
    TRISB = 0B11111100;      //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：  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：SPI_RW
 *   功能：    主机输出以及输入一个字节
 *   输入：    data
 *   输出：    根据接收的 data 输出给从机一个字节
 ----------------------------------------------*/
unchar SPI_RW(unchar data)
{

    unchar spiData;

    while(BUSY);                    //等待 SPI 模块空闲
    SPIDATA=data;
    while(BUSY || !SPIF);
    SPIF=0;
    spiData = SPIDATA;
    return spiData;
 }
/*----------------------------------------------
 *   函数名：WriteEnable
 *   功能：    写允许（将 WEN 置位）
 ----------------------------------------------*/
void WriteEnable(void)
{
    CS=0;
    SPI_RW(0x06);
    CS=1;
}
/*----------------------------------------------
 *   函数名：WriteDisable
```

```
 *   功能：    写禁止（将 WEN 复位）
 -----------------------------------------------*/
 void WriteDisable (void)
{
      CS=0;
      SPI_RW(0x04);
      CS=1;
}
/***********************************************
// 功能：读取 25C64 芯片的状态。
// 返回值：状态寄存器数据字节
// 注：25C64 内部状态寄存器第 0 位=0 表示空闲，0 位=1 表示忙。
***********************************************/
unchar SPI_ReadStatus(void)
{
      unchar status=0;
      CS=0;
      SPI_RW(0x05);                    //0x05 读取状态的命令字
      status = SPI_RW(0x00);
      CS=1;                            //关闭片选
      return status;
}
/***********************************************
*程序名：SPI_WriteStatus
*功能:   写 25C64 芯片的状态寄存器。
*         只有 BP1、BP0 (bit7、3、2)可以写、
*注：      25c64 内部状态寄存器第 0 位=0 表示空闲，0 位=1 表示忙。
***********************************************/
void SPI_WriteStatus(unchar Status)
{
      CS=0;
      SPI_RW(0X01);                    //0x01 读取状态的命令字
      SPI_RW(Status);                  //写入一个字节
      CS=1;                            //关闭片选
}
/*****************************************************************
程序名：SPI_Read
输入: 16 位的地址
返回: 读取的数据
说明：从 25c64 指定的地址读取一个字节
*****************************************************************/
unchar SPI_Read(unint addr)
{
      unchar spidata;
```

```
    while(SPI_ReadStatus()&0x01);          //判断是否忙
    CS=0;                                  //使能器件
    SPI_RW(0x03);                          //发送读取命令
    SPI_RW((unsigned char)((addr)>>8));
    SPI_RW((unsigned char)addr);
    spidata = SPI_RW(0x00);                //读出数据
    CS=1;
    return spidata;
}
/*********************************************************************
程序名：SPI_Write
输入: 地址，字节数据
说明：将一个字节写入指定的地址
*********************************************************************/
void SPI_Write(unint addr,unchar dat)
{
    while(SPI_ReadStatus()&0x01);          //判断是否忙
    WriteEnable();                         //SET WEL
    CS=0;                                  //使能器件
    SPI_RW(0x02);                          //发送写命令
    SPI_RW((unchar)((addr)>>8));
    SPI_RW((unchar)addr);

    SPI_RW(dat);
    WriteDisable();
    CS=1;                                  //关闭片选
    while(SPI_ReadStatus()&0x01);
}
/*---------------------------------------------
 *  函数名: main
 *  功能：  主函数
 *  输入：  无
 *  输出：  无
 ---------------------------------------------*/
void SPI_INITIAL(void)
{
    PCKEN |=0B00010000;          //开启 SPI 时钟

    SPICTRL    =0B00000000;   //
    //BIT7:传输完成标志
            //0：表示没有传输完成或者已经清零
            //1：传输完成标志位，写零清理，写 1 无效
        //BIT6:BUF 写入失败标识，
            //0：BUF 写入正常
```

```
            //1：BUF 为非空时，进行写入会置位该位，写零清理，写 1 无效
        //BIT5:工作模式错误标识,
            //0：工作模式正常
            //1：当 SPI 配置为主机模式，并且 NSS 用作输入引脚时，若 NSS 引进为低电平就会产生置位该位
            //写零清零，写 1 无效
        //BIT4:接收溢出标志
            //0：接收正常
            //1：接收溢出，写零清理，写 1 无效
        //BIT3~BIT2:NSS 引脚模式选择，当用作输出模式时，相应的 IO 脚会被用作 NSS 输出
            //00:禁用 NSS 引脚
            //01:NSS 引脚用作输入
            //1x:NSS 引脚用作输出，输出的值等于 NSSM[0]的值
            //注：NSS 引脚配置为输入状态时，可以被 SSM 的软件管理模式屏蔽掉
        //BIT1:发送 BUFF 为空状态
            //0：发送 BUF 非空
            //1：发送 BUF 位空
        //BIT0:SPI 接口使能
            //0：禁用 SPI 模块
            //1：启用 SPI 模块，相应的 IO 会被用作 SPI 的功能


    SPICFG      =0B01000000;    //
    //BIT7:SPI   BUSY 状态
            //0：SPI 模块空闲
            //1：表示 SPI 模块忙绿中
        //BIT6:MASTER 使能位
            //0：工作在 SLAVE 模式
            //1：工作在 MSATER 模式
        //BIT5:SCK 相位选择
            //0：第一个时钟转换的沿是数据采样点
            //1：第二个时钟转换的沿是数据采样点
        //BIT4:SCK 极性选择
            //0：SPI 空闲时，SCK 的时钟是处于低电平状态
            //1：SPI 空闲时，SCK 的时钟是处于高电平状态
        //BIT3:SLAVE 选择标志
            //0：该模块未被选中
            //1：该模块被选中
            //注：当 NSS 用作输入时，该值可以被 SSM 软件管理，当 SSM 为 1 时，这里的值表示的是 SSI 的值取反
        //BIT2:NSS 引脚的输入值状态
            //注：当 NSS 用作输入时，该值可以被 SSM 软件管理，当 SSM 为 1 时，这里的值表示的是 SSI 的值
```

```
        //BIT1:移位寄存器为空状态
            //0：内部串行移位寄存器非空
            //1：内部串行移位寄存器为空
        //BIT0:接受 BUFFER 为空状态
            //0：表示接收 BUF 非空
            //1：接收 BUF 为空状态

    SPISCR      =0B00000000;    //波特率设置,SPI 时钟 8M     16/2*(SPISCR+1)
    //SPICRCPOL =0B00000000; //CRC 计算多项式，使用默认值 0x07
    SPIRXCRC =0B00000000;       //接收数据的 CRC 计算结果
    SPITXCRC =0B00000000;       //发送数据的 CRC 计算结果
    SPIIER      =0B00000000;  //禁用 SPI 中断
    SPICTRL2    =0B00000000;
    //BIT7:半双工使能
        //0：禁用半双工模式
        //1：使能半双工模式
    //BIT6:半双工模式的接收使能
        //0：半双工模式接收使能
        //1：半双工模式发送使能
    //BIT5:全双工模式只允许接收使能
        //0：全双工模式允许发送和接收
        //1：全双工模式只允许接收
    //BIT4:NSS 输入管脚的值，仅当 SSM 置 1 时有效
        //0：输入到 NSS 引脚的值是 0
        //1：输入发哦 NSS 引脚的值是 1
    //BIT3:软件 SLAVE 模式管理，使能后 NSS 引脚的值由 SSI 替代
        //0：禁用 NSS 引脚的软件管理模式
        //1：启用 NSS 引脚的关键管理模式，如果 NSS 引脚用作输入，则 NSS 引脚实际的值
有 SSI 取代
    //BIT2:置位后在 TXBUFF 为空时会把 TXCRC 的值写入 TXBUFF
        //0：不传送 TXCRC 的值到 TXBUF
        //1：等待 TXBMT 为 1 时，传送 TXCRC 的值到 TXBUF，写入完成后改位自动清零
    //BIT1:CRC 模块计算使能
        //0：禁用 CRC 校验模块
        //1：启用 CRC 校验模块
    //BIT0:发送低比特位使能
        //0：高比特位优先发送
        //1：低比特位优先发送

    SPISTAT     =0B00000000;

    SPIEN=1;                    //使能 SPI

    //SPIF=0;   //1:传输完成标志位，写 0 清 0
```

```
    //BUSY        //1 表示 SPI 模块忙
    //RXBMT       //1 接收 BUF 为空状态
    //TXBMT       //1 发送 BUF 为空
    //SRXBMT      //1 表示 BUF 为空状态
}
/*-----------------------------------------------
 *  函数名: main
 *  功能：    主函数
 *  输入：    无
 *  输出：    无
 -----------------------------------------------*/
void main(void)
{
    SPIReadData=0;
    POWER_INITIAL();    //系统初始化
    SPI_INITIAL();    //SPI 初始化

    SPI_Write(0x0013,0x55); //写 0x13 地址
    SPI_Write(0x0014,0x88); //写 0x14 地址

    SPIReadData = SPI_Read(0x0013); //读取 0x13 地址值
    NOP();
    SPIReadData = SPI_Read(0x0014); //读取 0x14 地址值

    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/