FT64F0AX

SPI Application note

目录

FT64F0Ax SPI 应用

1. SPI 接口

SPI 接口可通过 SPI 协议与外部设备进行通信，特性如下：

- 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
- 输出支持 RZ 码调制

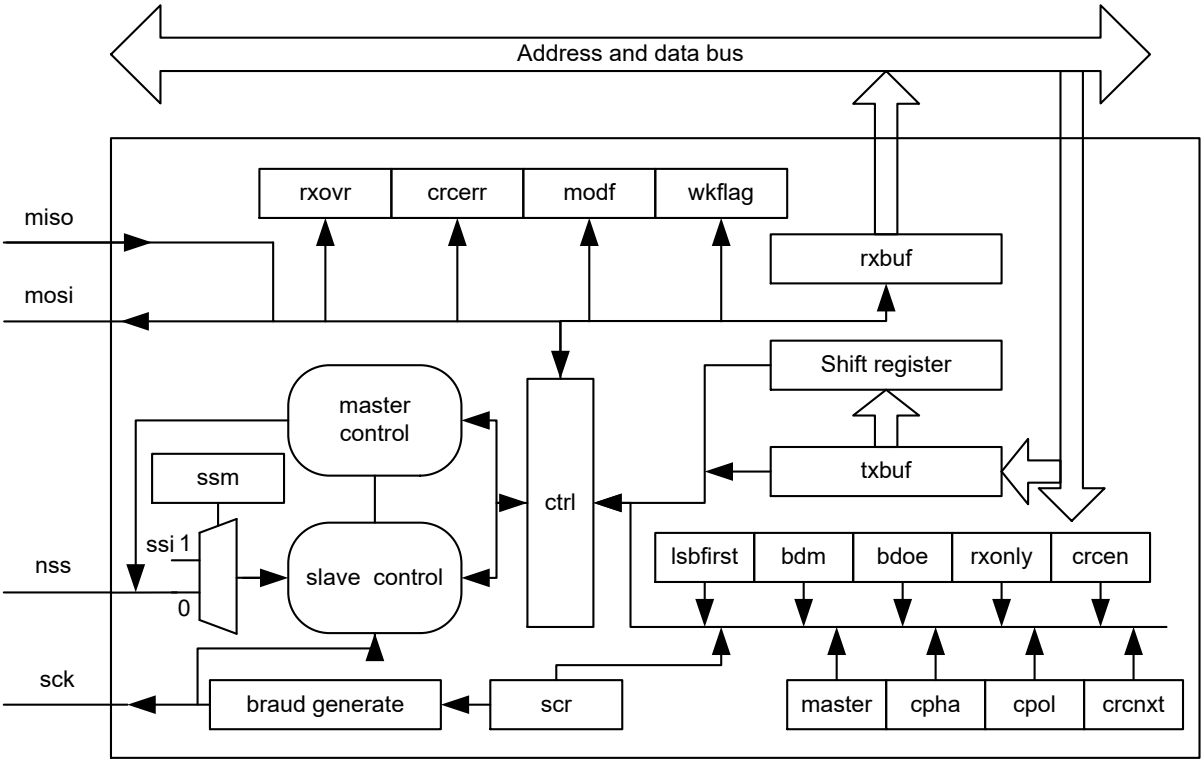

图 1-1 SPI 结构框图

SPI 接口有 4 个引脚：

名称	功能	主机模式	从机模式
MOSI	主机输出/从机输入	数据发送	数据接收
MISO	主机输入/从机输出	数据接收	数据发送
SCK	串行时钟	时钟输出	时钟输入
NSS	从机片选脚	–	输入，低电平有效

表 1-1 SPI 接口引脚说明

注:

1. 本章节中的 MOSI / MISO / SCK / NSS 分别对应引脚图中的 SPI_MOSI / SPI_MISO / SPI_SCK / SPI_NSS。
2. 从机片选 NSS 引脚配置：
   - NSS 引脚可配置成输入、输出或禁用三种状态 (参阅 ”NSSM”)；
   - NSS 用作输入时，其输入值 NSSVAL 为端口电平值(硬件)或 SSI 值(软件，参阅 ”SSM”)；
   - 从机模式下，只有当 NSS 使能输入且为低电平时，才能接收数据；
   - 主机模式下，如果 NSS 使能输入且为低电平，则会导致工作模式错误(置位 MODF)，此时 SPI 模块自动切换至从机模式，此特性可用于兼容多主机通信；

SPI 接口支持全双工(四线/三线)和半双工(二线)同步数据传输，默认为全双工，SPI 通信总是由主机发起。

全双工模式，在同一时钟信号 (主机输出的串行时钟) 下，数据输出和数据输入同步进行。半双工模式下，主机模式的数据脚为 MOSI，从机模式的数据脚为 MISO。

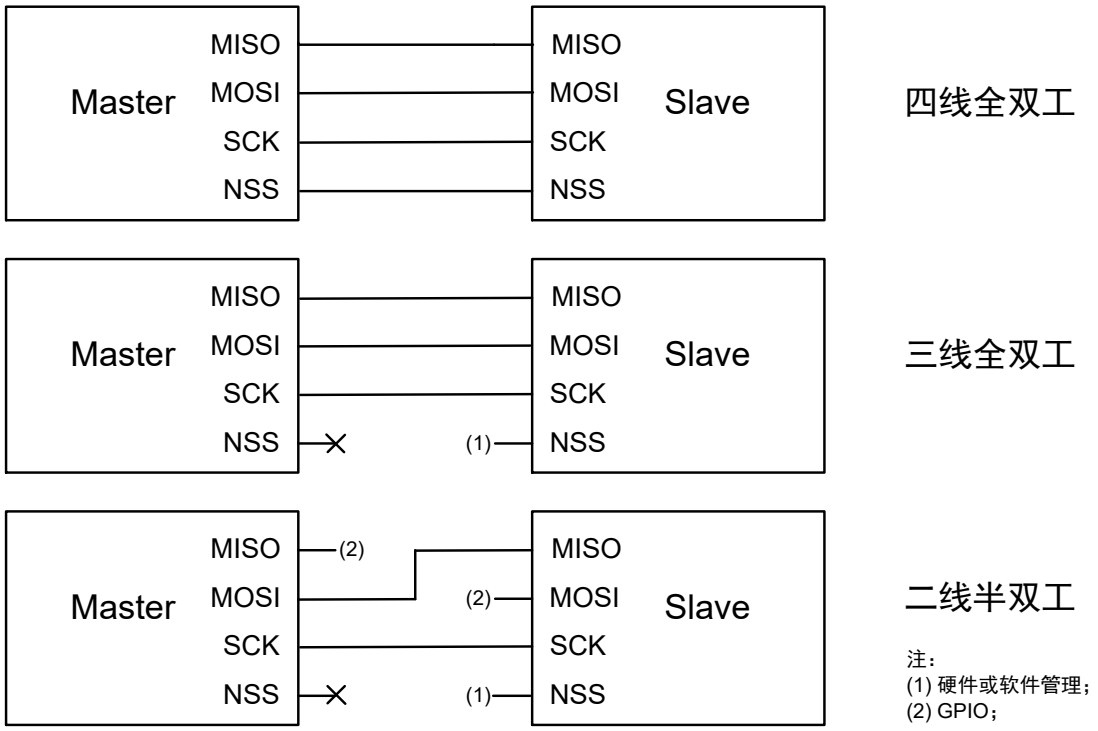

图 1-2 SPI 接口引脚连接示意图

1.1. SPI 相关寄存器汇总

名称	状态	寄存器	地址	复位值
DATA	数据发送/接收 BUF (TXBUF/RXBUF) 写时：将新数据写入到 TXBUF 中 读时：返回 RXBUF 中未读的数据	SPIDATA[7:0]	0x15	RW-0000 0000
SPIF [1]	数据传输完成标志 1 = 完成 (锁存) 0 = 未完成，或已被清零	SPICTRL[7]	0x16	RW0-0
WCOL [1]	BUF 写入失败(非空时写入)标志 1 = 失败 (锁存) 0 = 正常	SPICTRL[6]		RW0-0
NSSM	NSS 引脚模式选择 00 = 禁用 01 = 输入 (输入值 NSSVAL 与 SSM，端口电平及 SSI 有关) 1x = 输出 (输出值 = NSSM[0])	SPICTRL[3:2]		RW-01
SPIEN	SPI 接口 1 = 使能 0 = 关闭	SPICTRL[0]		RW-0
SBUSY	SPI 状态 1 = 忙碌中 0 = 空闲	SPISTAT[4]	0x1E	RO-0
RZMEN	RZ 调制模式 1 = 使能 0 = 禁用	SPICFG[7]	0x17	RW-0
MSTEN	工作模式 1 = 主机模式 (MASTER) 0 = 从机模式 (SLAVE)	SPICFG[6]		RW-0
CPHA	SCK 相位选择 (数据采样点) 1 = 第 2 个时钟转换沿 0 = 第 1 个时钟转换沿	SPICFG[5]		RW-0
CPOL	SCK 极性选择 (SPI 空闲时，SCK 时钟状态) 1 = 高电平 0 = 低电平	SPICFG[4]		RW-0
SLAS	从机选中标志位 1 = 被选中 0 = 未被选中	SPICFG[3]		RO-0
NSSVAL	NSS 引脚输入值 当 SSM=0 时，NSSVAL = NSS 引脚端口电平值 当 SSM=1 时，NSSVAL = SSI	SPICFG[2]		RO-1

[1] 写 0 清零，写 1 无效。

名称	状态	寄存器	地址	复位值
SRMT	内部串行移位寄存器状态 1 = 空 0 = 非空	SPICFG[1]		RO-1
SPICKEN	SPI 模块时钟 1 = 使能 0 = 关闭	PCKEN[4]	0x9A	RW-0
SYSON	睡眠模式下，系统时钟控制 1 = 保持运行 0 = 关闭	CKOCON[7]	0x95	RW-0
SCR	SCK 速率设置 (仅主机模式有效) 速率 = Fmaster/(2*(SCR+1)) (SPI 外设时钟 Fmaster = Sysclk)	SPISCR[7:0]	0x18	RW-0000 0000
BDM	半双工 1 = 使能 0 = 关闭	SPICTRL2[7]	0x1D	RW-0
BDOE	半双工工作模式 1 = 发送 0 = 接收	SPICTRL2[6]		RW-0
RXONLY	全双工工作模式 1 = 只允许接收 0 = 允许发送和接收	SPICTRL2[5]		RW-0
SSI	NSS 引脚软件输入值 (仅当 SSM = 1 时有效) 1 = 输入值为 1 0 = 输入值为 0	SPICTRL2[4]		RW-0
SSM	从机模式下，NSS 引脚输入值管理 1 = 软件 0 = 硬件	SPICTRL2[3]		RW-0
CRCNXT	发送 TXCRC 值到 TXBUF 1 = 发送 (完成后自动清零) 0 = 不发送	SPICTRL2[2]		RW-0
CRCEN	硬件 CRC 校验模块 1 = 关闭 0 = 使能	SPICTRL2[1]		RW-0
LSBFIRST	数据传输格式 1 = 优先发送低比特位 (LSB) 0 = 优先发送高比特位 (MSB)	SPICTRL2[0]		RW-0
CRCPOL	CRC 计算多项式 (默认值: 0x07)	SPICRCPOL[7:0]	0x19	RW-0000 0111
RXCRC	接收数据的 CRC 计算结果 (CRCEN 由 0 变 1，此位自动清零)	SPIRXCRC[7:0]	0x1A	RO-0000 0000
TXCRC	发送数据的 CRC 计算结果 (CRCEN 由 0 变 1，此位自动清零)	SPITXCRC[7:0]	0x1B	RO-0000 0000

表 1-3 SPI 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, TXE, RXNE, RXERR, WAKUP 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 (TXE, RXNE, RXERR, WAKUP 适用) 0 = 关闭 (无唤醒)		INTCON[6]		RW-0
TXE	发送 BUF 为空中断	1 = 使能 0 = 关闭 (无唤醒)	SPIIER[0]	0x1C	RW-0
TXBMT	发送 BUF 状态位	1 = 空 0 = 非空	SPICTRL[1]	0x16	RO-1
STXBMT			SPISTAT[2]	0x1E	RO-1
RXNE	接收 BUF 为非空中断	1 = 使能 0 = 关闭 (无唤醒)	SPIIER[1]	0x1C	RW-0
RXBMT	接收 BUF 状态位	1 = 空 0 = 非空	SPICFG[0]	0x17	RO-1
SRXBMT			SPISTAT[3]	0x1E	RO-1
RXERR	接收错误中断 (工作模式错误，接收溢出，CRC 校验错误)	1 = 使能 0 = 关闭 (无唤醒)	SPIIER[2]	0x1C	RW-0
MODF [2]	工作模式错误标志位 1 = 错误 (锁存) (主机模式下，NSS 脚使能输入且为低电平，导致模式错误) 0 = 正常		SPICTRL[5]	0x16	RW0-0
SMODF			SPISTAT[6]	0x1E	RO-0
RXOVRN [2]	接收溢出标志位	1 = 溢出 (锁存) 0 = 正常	SPICTRL[4]	0x16	RW0-0
SRXOVRN			SPISTAT[5]	0x1E	RO-0
CRCERR [2]	CRC 校验错误标志位	1 = 错误 (锁存) 0 = 正确，或已被清零	SPISTAT[0]	0x1E	RW0-0
WAKUP	从机唤醒中断	1 = 使能 0 = 关闭	SPIIER[3]	0x1C	RW-0
WKF [2]	从机唤醒(接收到数据)标志位	1 = 已唤醒 (锁存) 0 = 未唤醒，或已被清零	SPISTAT[1]	0x1E	RW0-0

表 1-4 SPI 中断使能和状态位

[2] 写 0 清零，写 1 无效。

名称	状态		寄存器	地址	复位值
AFP1[2]	SPI_SCK	1 = PB2 0 = PB0	AFP1[2]	0x19F	RW-0
AFP1[1]	SPI_MOSI	1 = PB7 0 = PA0	AFP1[1]		RW-0
AFP1[0]	SPI_MISO	1 = PC1 0 = PA1	AFP1[0]		RW-0
SPIOD	SPI_MISO，SPI_MOSI 开漏输出 1 = 使能 0 = 关闭		ODCON0[2]	0x21F	RW-0

表 1-5 SPI 接口引脚控制

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
SPIDATA	015h	DATA[7:0]								0000 0000
SPICTRL	016h	SPIF	WCOF	MODF	RXOVRN	NSSM		TXBMT	SPIEN	0000 0010
SPICFG	017h	RZMEN	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	---0 0000
SPICTRL2	01Dh	BDM	BDOE	RXONLY	SSI	SSM	CRCNXT	CRCEN	LSBFIRST	0000 0000
SPISTAT	01Eh	—	SMODF	SRXOVRN	SBUSY	SRXBMT	STXBMT	WKF	CRCERR	-000 0000

表 1-2 SPI 相关寄存器地址

1.2. SPI 配置

主机和从机的 SPI 配置流程基本相同：

1. 使能 SPI 模块时钟（参阅” SPICKEN”）；
2. 选择主机或从机模式（参阅” MSTEN”）；
3. 配置 NSS 引脚（参阅” NSSM”, ” SSM”, ” SSI”和” NSSVAL”）；
4. SCK 通信速率，主机模式的速率可设置为 Fmaster/(2*(SCR+1))，从机模式的速率高达 Fmaster/4；
5. 设置 SCK 的相位和极性（参阅” CPOL”和” CPHA”）；
6. 选择数据传输格式（参阅” LSBFIRST”）；
7. 设置全双工（参阅”RXONLY”）或半双工工作模式（参阅” BDM”和” BDOE”）；

8. 如需要，可使能硬件 CRC 校验模块（参阅” CRCPOL”和” CRCEN”）；
9. 使能 SPI 模块（参阅” SPIEN”）；
10. 如需要，可使能相应的中断（参阅” GIE”, ” PEIE”, ” RXERR” , ” RXNE” , ” TXE”和” WAKUP”）；

注：

- SPI 外设时钟 Fmaster = Sysclk；
- SPI 使能时，引脚 MOSI / MISO / SCK / NSS 接口功能自动使能；
- 主机发送 SCK 时钟之前，需要先使能 SPI 从机；
- 主机作为发送端时，SPI 使能且 TXBUF 为非空时，主机自动发起传输；
- 主机作为只接收模式（RXONLY=1 或 BDM=1& BDOE=0）时，SPI 使能后，主机自动发起传输并一直发送 SCK；
- 主机发起传输之前，从机的数据寄存器中需提前写入需发送的数据（连续通信时，需要在正在进行中的传输结束之前继续向从机的数据寄存器写入数据）；
- 当 SPIEN 由 0 变 1 时，SPIF / MODF / RXOVRN / CRCERR / WKF 自动清零，TXBMT / RXBMT 自动置位；

1.2.1. 通信时钟 SCK 设置

时钟 SCK 的极性和相位可配置为图 1-3 所示的 4 种情况（参阅” CPOL”, ” CPHA”）。

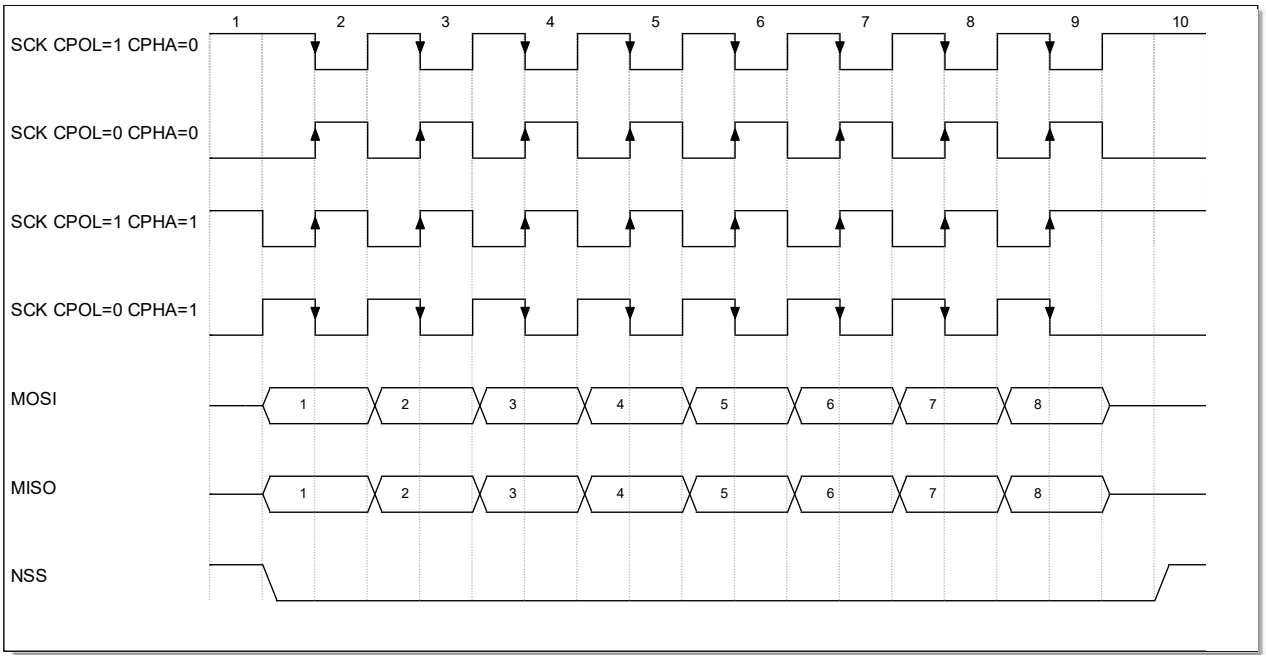

图 1-3 SCK 时钟极性和相位时序图

1.2.2. 数据处理流程

数据通信流程分为阻塞模式和非阻塞模式，处理方式一致，区别在于非阻塞模式在中断中进行。

流程	阻塞模式	非阻塞模式
发送数据	向 DATA(TXBUF)写入数据后，查询 TXBMT，当其置 1 时，写入下一个数据	向 DATA(TXBUF)写入数据后，当 TXE = 1 时，TXBMT 置 1 后会进入中断
接收数据	查询 RXBMT，当其为 0 时，则可读取 DATA(RXBUF)的值	当 RXNE = 1 时，RXBMT 置 0 则会进入中断
	查询 RXOVRN 和 CRCERR，当 RXOVRN 或 CRCERR 置 1 时，需软件清零相应的错误标志位	当 RXERR = 1 时，RXOVRN 或 CRCERR 置 1 后会进入中断 (需软件清零相应的错误标志位)
备注	-	进入中断后，查询相应的状态标志位并处理发送接收流程，处理完成后退出中断

表 1-6 SPI 数据处理流程

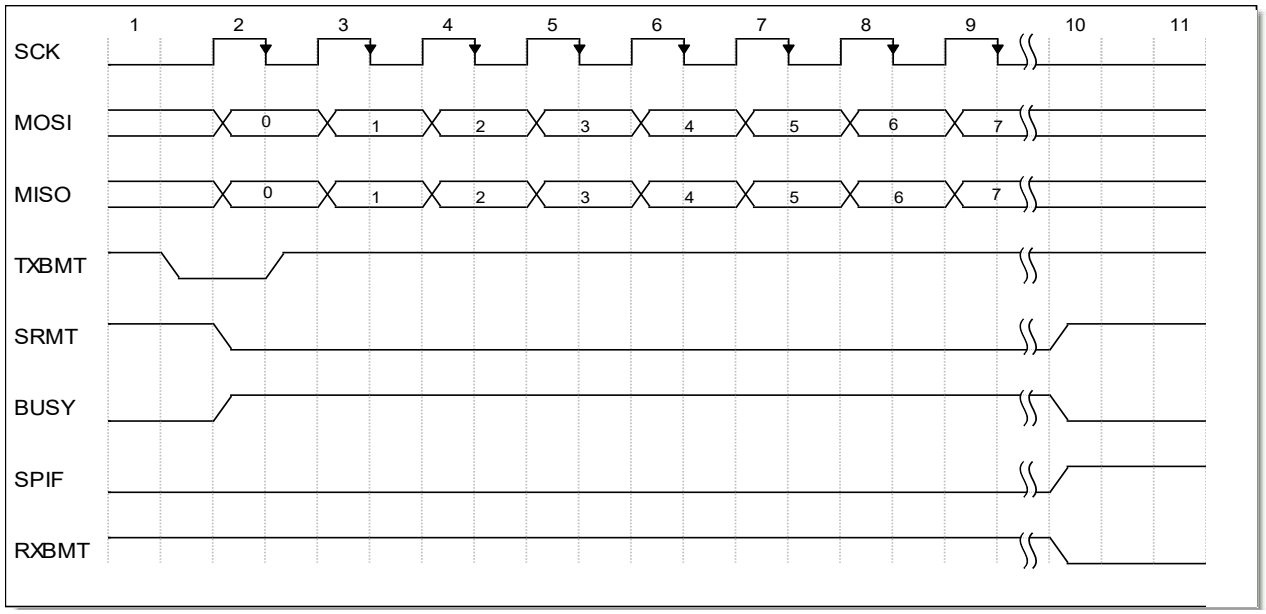

图 1-4 数据处理时序图 (以单字节数据传输为例)

以全双工通信流程为例，无论阻塞模式还是非阻塞模式，通信过程中的相关标志位变化如图 1-4 所示：

1. 向 DATA 寄存器写入数据后，TXBMT 由 1 变为 0；
2. TXBUF 中的数据传送到内部移位寄存器，SRMT 由 1 变为 0；
3. 移位寄存器中的数据完全移出后，SRMT 由 0 变为 1；
4. 发送过程中，BUSY 一直为 1；
5. 当前字节数据传输完成后，SPIF 由 0 变为 1，同时 RXBMT 由 1 变为 0，此时可读取 DATA 寄存器的值；

注： 全双工或半双工模式下，需在完成发送/接收全部数据(TXBMT=1 / RXBMT=0)后，且 SPI 处于空闲状态(BUSY=0)时，才能关闭 SPI 模块；

1.2.3. 硬件 CRC 校验

CRC 校验模块用于增强数据传输的可靠性。

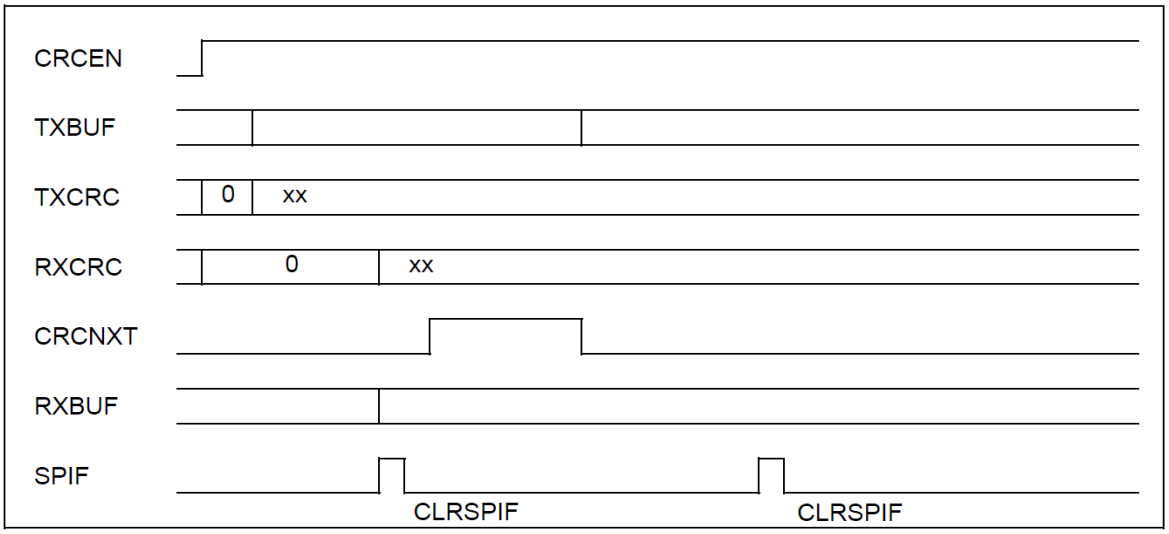

图 1-5 CRC 模块的工作时序图

配置 CRCEN = 1，使能硬件 CRC 校验模块：

- 发送端：
  1. 每次正常写入到 TXBUF 的值，会被送到 CRC 模块，连同多项式 CRCPOL 生成 TXCRC 的值；
  2. 当正常数据全部发送完成后，配置 CRCNXT = 1，下一次传输时将自动发送最后的 CRC 校验码值，即自动将 TXCRC 值写入到 TXBUF(本次写入到 TXBUF 中的值不会再被送到 CRC 模块进行计算)，CRCNXT 的值自动清零；
- 接收端：
  1. 每次正常写入到 RXBUF 的值，会被送到 CRC 模块，连同多项式 CRCPOL 生成 RXCRC 的值；
  2. 当正常数据全部接收完成后，下一次将自动接收对方的 CRC 检验码值（本次接收到的数据不会再写入到 RXBUF）并与 RXCRC 值进行比较，如果不匹配则会置位 CRCERR

注： 当 CRCEN 由 0 变 1 时，会对 CRC 模块进行初始化（TXCRC 和 RXCRC 被清零），但不影响 CRC 计算多项式 CRCPOL 的值（默认为 0x07）。

CRC 校验码值传输同样分为阻塞模式和非阻塞模式：

流程	阻塞模式	非阻塞模式
发送 CRC 校验码	当最后一个数据传输完成时： 1. 查询 TXBMT，当其置 1 时则置位 CRCNXT； 2. 查询 CRCNXT，当其为 0 时清零 SPIF； 3. 查询 SPIF，当其置 1 时表示 CRC 校验码发送完成；	当 TXE = 1 时，TXBMT 置 1 会进入中断，如果数据已经发送完整，则软件置位 CRCNXT；
接收 CRC 校验码	查询 CRCERR，当其为 1 时，表示 CRC 校验码不匹配，需软件清零相应的标志位	当 RXERR = 1 时，CRCERR 置 1 会进入中断（需软件清零相应的标志位）

表 1-2 CRC 校验码处理流程

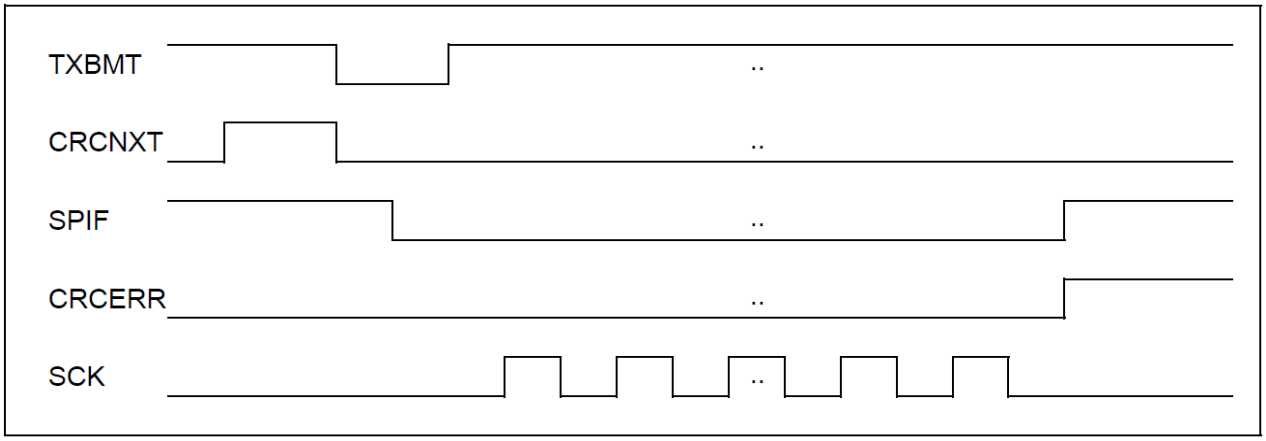

图 1-6 CRC 模块标志位时序图

1.2.4. 从机模式的睡眠唤醒

睡眠模式下，如果 SPICKEN、SYSON、WAKEUP、PEIE 同时使能，从机在接收到第 1 个比特数据时，即可唤醒 MCU。

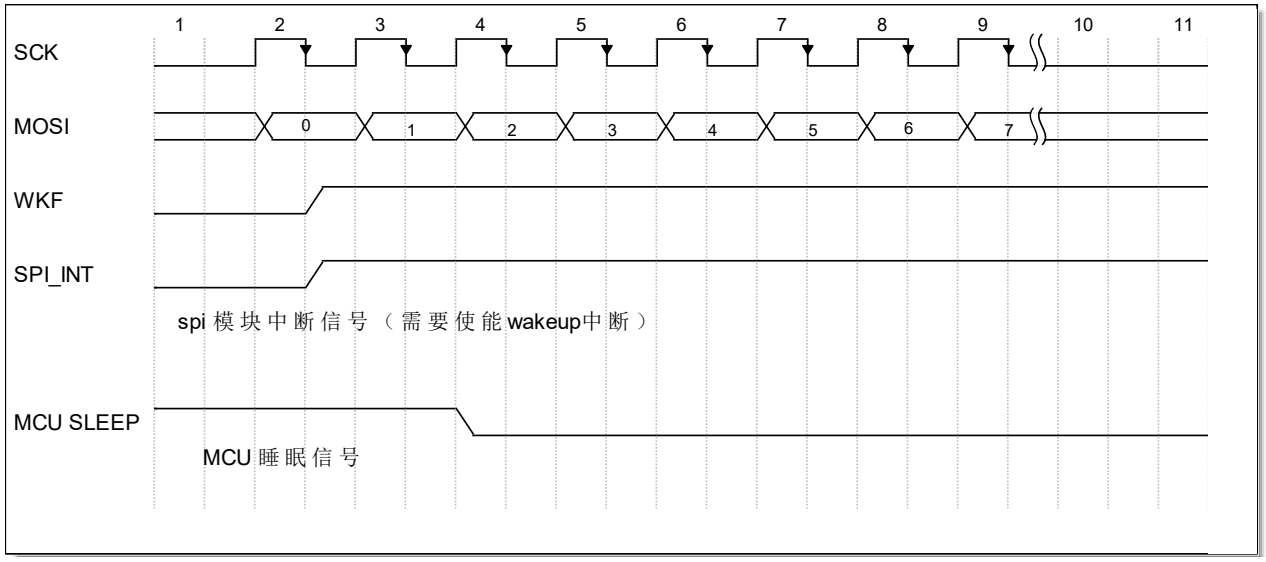

图 1-7 睡眠唤醒时序图

1.2.5. RZ 码调制

当 RZMEN 位为 1 时，输出数据将被为调制成归零码，其中数据“1”的占空比是 2/3，数据“0”占空比是 1/3。

RZ 调制只支持单工通信，即主机发送模式 (不支持主机接收)，跟普通主机模式不一样，该模式下的 SCK 和 MISO 将不起作用，它们将作为普通 IO 使用。

RZ 模式下的波特率时钟由下式决定，其中 Fmaster 为系统时钟：

Fbaud = Fmaster / (SCR + 1)，tE = 1 / Fbaud

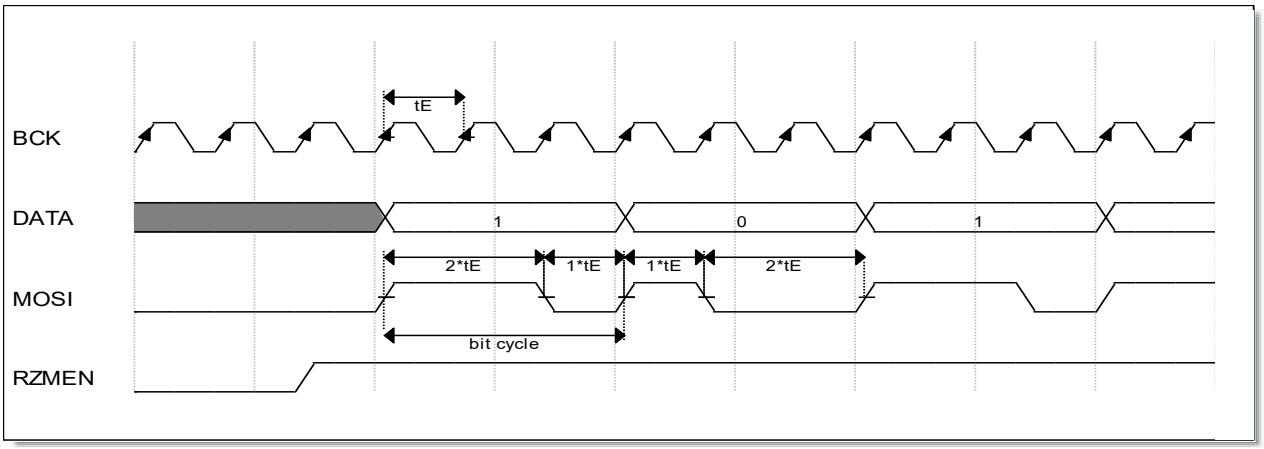

图 1-8 RZ 码调制时序图

2. 应用范例

```
//*************************************************************
/* 文件名：TEST_64F0Ax_SPI.c
* 功能：    FT64F0Ax_SPI 功能演示
*  IC：     FT64F0A5   TSSOP20
* 内部：    16M
* 说明：    此演示程序为 64F0Ax_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
*
*                   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|--------MISO
* NC----------|5(PC1)       (PA0)16|--------MOSI
* NC----------|6(PB7)       (PB0)15|---------SCK
* GND---------|7(GND)       (PB1)14|----------NC
* NC----------|8(PB6)       (PB2)13|----------NC
* VDD---------|9(VDD)       (PB3)12|----------NC
* NSS---------|10(PB5)      (PB4)11|----------NC
*              ------------------------------
*/
//*************************************************************
#include      "SYSCFG.h";
#include      "FT64F0AX.h";
//**********************宏定义*****************************
#define  uchar    unsigned char
#define  uint     unsigned int

#define       NSS          NSSM0

volatile  uchar    SPIReadData=0;
volatile  uchar    SPIReadData1=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;              //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=0B00000010;                //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=0B00000010;           //输入输出设置，0-输出，1-输入，SPI_MISO,SPI_MOSI
    TRISB=0B00000000;               //SPI_NSS,SPI_SCK   PB0,PB5-OUT
    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 SPI
* 输入：   无
* 输出：   无
---------------------------------------------*/
void SPI_INITIAL(void)
{
    PCKEN|=0B00010000;              //使能 SPI 模块时钟

    SPICTRL=0B00001000;             //NSS 引脚输出
    SPICFG=0B01000000;              //设置为主机模式，第一个时钟转换的沿是数据采样点，SPI 空
闲时，SCK 的时钟是处于低电平状态
    SPISCR=0B00000000;              //波特率设置为 8M，Fmaster/(2*(SPISCR+1))，Fmaster=16M
    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;            //禁止所有中断
    SPICTRL2=0B00000000;          //全双工允许发送和接收，禁用 CRC 校验模块，高比特位优先
发送
    SPISTAT=0B00000000;
    SPIEN=1;                      //启用 SPI
}
/*---------------------------------------------
* 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
---------------------------------------------*/
uchar SPI_RW(uchar data)
{
    uchar spiData;

    while(SBUSY);                 //等待 SPI 模块空闲
    SPIDATA=data;
    while(RXBMT);
    spiData=SPIDATA;
    return spiData;
}
/*---------------------------------------------
* 函数名：WriteEnable
* 功能：   写允许（将 WEN 置位）
---------------------------------------------*/
void WriteEnable(void)
{
    NSS=0;
    SPI_RW(0X06);
    NSS=1;
}
/*---------------------------------------------
* 函数名：WriteDisable
* 功能：   写禁止（将 WEN 复位）
---------------------------------------------*/
void WriteDisable(void)
{
    NSS=0;
    SPI_RW(0X04);
    NSS=1;
}
```

```
/*-------------------------------------------------
 * 函数名：SPI_ReadStatus
 * 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。
 -------------------------------------------------*/
 uchar SPI_ReadStatus(void)
 {
      uchar status=0;
      NSS=0;
      SPI_RW(0X05);                  //0x05 读取状态的命令字
      status=SPI_RW(0X00);
      NSS=1;                         //关闭片选
      return status;
 }
/*-------------------------------------------------
 * 函数名：SPI_WriteStatus
 * 功能：   写 25C64 芯片的状态寄存器
 * 注：     只有 BP1、BP0(bit7、3、2)可以写
            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(uchar Status)
 {
      NSS=0;
      SPI_RW(0X01);                  //0X01 写入状态的命令字
      SPI_RW(Status);                //写入一个字节
      NSS=1;                         //关闭片选
 }
/*-------------------------------------------------
 * 函数名：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
 -------------------------------------------------*/
 uchar SPI_Read(uint addr)
 {
      uchar spidata;
      while(SPI_ReadStatus()&0x01);  //判断是否忙
      NSS=0;                         //使能器件
      SPI_RW(0X03);                  //发送读取命令
      SPI_RW((uchar)((addr)>>8));
      SPI_RW((uchar)addr);
      spidata=SPI_RW(0X00);          //读出数据
      NSS=1;
```

```
    return spidata;
}
/*-------------------------------------------------
 * 函数名：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(uint addr , uchar dat)
{
    while(SPI_ReadStatus()&0x01);    //判断是否忙
    WriteEnable();                   //置位 WEN
    NSS=0;                           //使能器件
    SPI_RW(0X02);                    //发送写命令
    SPI_RW((uchar)((addr)>>8));
    SPI_RW((uchar)addr);

    SPI_RW(dat);
    NSS=1;                           //关闭片选
    WriteDisable();
    while(SPI_ReadStatus()&0x01);
}
/*-------------------------------------------------
 * 函数名：main
 * 功能：   主函数
 * 输入：   无
 * 输出：   无
 -------------------------------------------------*/
void main(void)
{
    SPIReadData=0;
    SPIReadData1=0;
    POWER_INITIAL();                 //系统初始化
    SPI_INITIAL();                   //SPI 初始化

    SPI_Write(0x13,0x55);            //写 0x13 地址
    SPI_Write(0x14,0x88);            //写 0x14 地址
    SPIReadData=SPI_Read(0x13);      //读取 0x13 地址值
    NOP();
    SPIReadData1=SPI_Read(0x14);     //读取 0x14 地址值
    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