FT61F02X

SPI Application note

Table of contents

FT61F02x SPIapplication

1. SPIRelated register settings

SPIis the Serial Peripheral Interface (Serial Peripheral Interface)abbreviation of.SPI, is a high-speed, full-duplex, synchronous communication bus that works in a master-slave mode. This mode usually has a master device and one or more slave devices, requiring at least4root line, in fact3 Root works too (when transferring in one direction). is also all based onSPIcommon to all devices, they are:

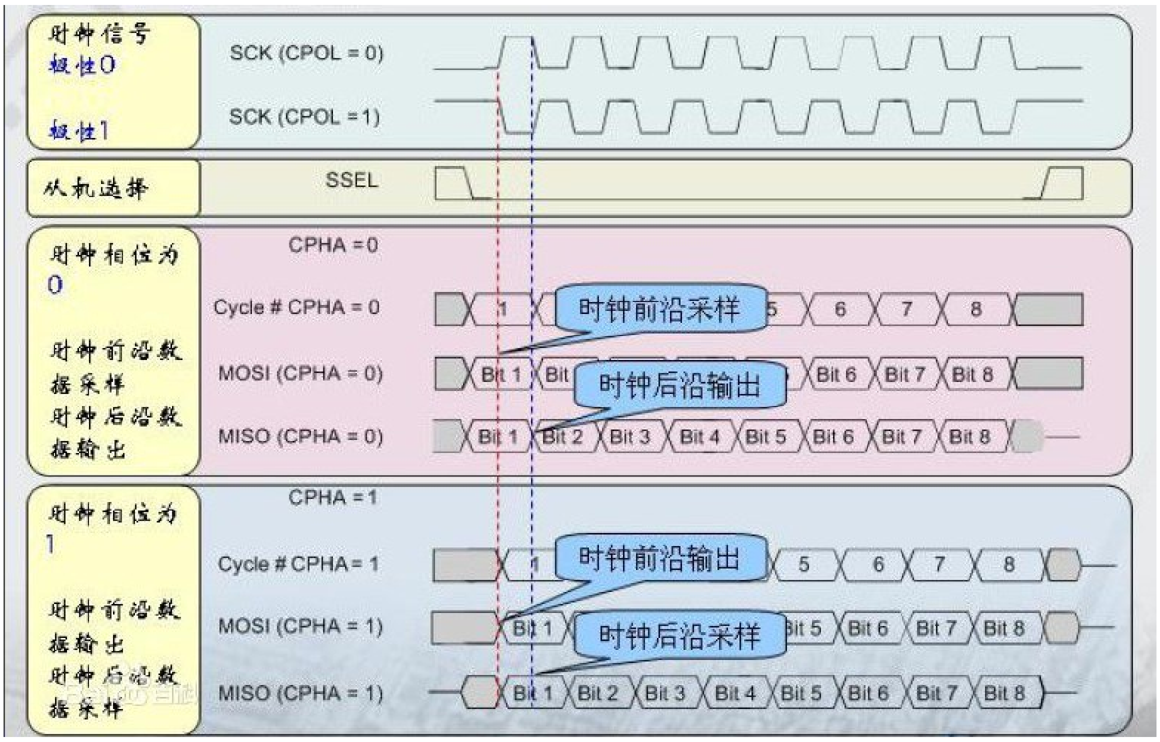
SDO/MOSI –-------Master device data output, slave device data input;

SDI/MISO –-------Master device data input, slave device data output;

SCLK ------------Clock signal, generated by the master device;

CS ---------------Chip select, slave device enable signal, controlled by the master device.

SPIcommunication has4Different slave devices may be configured in a certain mode at the factory, which cannot be changed; but our communication parties must work in the same mode, so we can configure our master deviceSPImode is configured by CPOL(clock polarity) and CPHA(clock phase) to control the communication mode of our master device



Mode0:CPOL=0,CPHA=0

Mode1:CPOL=0,CPHA=1

Mode2:CPOL=1,CPHA=0

Mode3:CPOL=1,CPHA=1

This program usesmode0working mode, toFT61F023 SOP16For example, the four data lines correspond toIOPin:

```
#define   MISO   PORTA,4

#define   MOSI   PORTA,2

#define   SCK    PORTA,6

#define   CS     PORTA,7
```

2.Application example

```
//****************************************************
********** /*file name:TEST_61F02x_SPI.c
* Function:     FT61F02x-SPIDemo
*IC:            FT61F023 SOP16
* Crystal:      16M/4T
* illustrate:   This demo program is61F23x_SPIdemo program for . The program reads (25C64)0x12The
*                value of the address, reversed and stored0x13address
*
*                     FT61F023 SOP16
*                    ----------------------------
* VDD----------|1(VDD)      (VSS)16|----------GND
* CS------------|2(PA7)      (PA0)15|------------NC
* SCK----------|3(PA6)       (PA1)14|------------NC
* NC------------|4(PA5)      (PA2)13|---------MOSI
* NC------------|5(PC3)      (PA3)12|------------NC
* NC------------|6(PC2)      (PC0)11|------------NC
* MISO---------|7(PA4)      (PC1)10|------------NC
* NC------------|8(PC5)      (PC4)09|------------NC
*                    ----------------------------
*/
//****************************************************** ***********
# include "SYSCFG.h"
//************************Macro definition **************************
#define   unchar  unsigned   char
#define   unint   unsigned   int

#define   MISO    PA4
#define   MOSI    PA2
#define   SCK     PA6
#define   CS      PA7

unchar SPIReadData;
 /*------------------------------------------------ -
* Function name:POWER_INITIAL
* Function:   Power-on system initialization
* enter:    none
* output:   none
 ------------------------------------------------
--------- */ void POWER_INITIAL (void)
{
      OSCCON = 0B01110001;           //IRCF=111=16MHz/4T=4MHz,0.25μs //Temporarily
      INTCON = 0;                    disable all interrupts
```

```
    PORTA = 0B00000000;
    TRISA = 0B00010000;              //PAinput Output0-output1-enterPA4-enter
    PORTC = 0B00000000;
    TRISC = 0B00000000;              //PCinput Output0-output1-enter //PAPort pull-
    WPUA = 0B00010000;               up control1-pull up0-close pull //PCPort pull-up
    WPUC = 0B00000000;               control1-pull up0-close pull

    ANSEL = 0B00000000;
    OPTION = 0B00001000;             //Bit3=1, WDT MODE, PS=000=WDT RATE 1:1
    MSCKCON = 0B00000000;
    //Bit6->0,prohibitPA4,PC5Regulated output
    //Bit5->0,TIMER2the clock isFosc //Bit4->0,
    prohibitLVR
    CMCON0 = 0B00000111;             //turn off the comparator,Cxfor numbersIOmouth
}

/*------------------------------------------------ -
* Function name:init_25c64_io
* Function:   25C64initialization
* enter:      none
* output:     none
  --------------------------------------------------
----- */ void init_25c64_io(void)
{
    CS = 1;
    SCK = 0;
    MOSI = 0;
}
/*------------------------------------------------ -
* Function name:SPI_RW
* Function:   The host outputs and inputs a byte
* enter:      data
* output:     received according todataoutput a byte to the slave
  --------------------------------------------------
--------- */ unchar SPI_RW(unchar data)
{
   unchar i;
   for(i=0;i<8;i++)
   {
    if(data&0x80)
       MOSI = 1;
    else
       MOSI = 0;
    NOP();
```

```
        data<<=1;
        SCK =       1;
        NOP();
        if(MISO)
            data   |= 0x01;
        else
            data   &= 0xFE;
        NOP();
        SCK =       0;
      }
      return data;
  }
/*------------------------------------------------ -
* Function name:WriteEnable
* Function: Write enable (willWenPosition)
  ---------------------------------------------------

- - - - - */ void WriteEnable(void)
{
        CS=0;
        SPI_RW(0x06);
        CS=1;
}
/*------------------------------------------------ -
* Function name:WriteDisable
* Function: Write Inhibit (WillWenreset)
  ---------------------------------------------------

  - - - - - * / void WriteDisable (void)
{
        CS=0;
        SPI_RW(0x04);
        CS=1;
}
/*------------------------------------------------
* Function name:SPI_ReadStatus
* Function: read25C64state of the chip.
* Return value: status register data byte
* Note:       25C64Internal Status Register No.0bit=0means free,0bit=1Indicates busy.
---------------------------------------------------

- - - - - - - - - */ unchar SPI_ReadStatus(void)
{
        unchar status=0;
        CS=0;
        SPI_RW(0x05);                              //0x05Command word to read status
        status = SPI_RW(0x00);
```

```
    CS=1;                              //Close Chip Select
    return status;
}
/*-----------------------------------------------
* Function name:SPI_WriteStatus
* Function:   Write25C64Chip status register.
*             onlyBP1,BP0 (bit7,3,2)can write,
* Note:       25c64Internal Status Register No.0bit=0means free,0bit=1Indicates busy.
------------------------------------------------

---------*/ void SPI_WriteStatus(unchar Status)
{
    CS=0;
    SPI_RW(0X01);                      //0x01Command word to read status
    SPI_RW(Status);                    //write a byte
    CS=1;                              //Close Chip Select
}
/*----------------------------------------------- -
* Function name:SPI_Read
* enter:      16bit address
* return:     read data
* illustrate: from25c64Read a byte from the specified address
------------------------------------------------

---------*/ unchar SPI_Read(unint addr)
{
    unchar spidata;
    while(SPI_ReadStatus()&0x01);      //determine if busy
    CS=0;                              //enable device
    SPI_RW(0x03);                      //send read command
    SPI_RW((unsigned char)((addr)>>8));
    SPI_RW((unsigned char)addr); spidata
    = SPI_RW(0x00);                    //read data
    CS=1;
    return spidata
}
/*----------------------------------------------- ----
* Function name:SPI_Write
* enter:      address, byte data
* illustrate: write a byte to the specified address
---------------------------------------------------

---------* / void SPI_Write(unint addr,unchar dat)
{
    while(SPI_ReadStatus()&0x01);      //determine if busy
    WriteEnable();                     //PositionWEL
    CS=0;                              //enable device
    SPI_RW(0x02);                                    //send write command
    SPI_RW((unchar)((addr)>>8));
    SPI_RW((unchar)addr);

    SPI_RW(dat);
    CS=1;                                            //Close Chip Select
    WriteDisable();
    while(SPI_ReadStatus()&0x01);
}
/*------------------------------------------------ -
* Function name:main
* Function:     main function
* enter:        none
* output:       none
 ---------------------------------------------------------

*/ void main()
{
    POWER_INITIAL();                                 //system initialization
    init_25c64_io();
    SPIReadData = SPI_Read(0x0012);                  //read0x12addressEEPROMvalue //Negate
    SPI_Write(0x0013,~SPIReadData);                  the write address0x13
    while(1)
    {
        NOP();
    }
}
```

contact information

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, Veristong 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