十速科技股份有限公司
tenx technology inc.

Advance Information

TM89 series

The implementation of SPI interface in C language

Application Note

Tenx reserves the right to change or discontinue this product without notice.

tenx technology inc.

CONTENTS

PRODUCT NAME **2**
TM89 系列IC **2**
APPLICATION NOTE **2**
一. SPI通信基本模組， 主要包括如下#define以及函數. **2**
1. #define部分說明 2
2. 函數部分說明 2
二. 讀寫M2510（SPI介面EEPROM）的#define以及函數。 **3**
1. #define部分說明 3
三. 主要函數詳細介紹 **3**
1. 函數 F_Sim_Spi_Write_bit 3
2. 函數F_Sim_Spi_Read_bit 4

PRODUCT NAME

TM89 系列 IC

TITLE

The implementation of SPI interface in C language

APPLICATION NOTE

本篇 AP 主要講述如何在 TM89 系列 IC 中使用 C 語言完成 SPI 串列通信功能，本 AP 的例副程式分爲兩個部分，第一部分爲 SPI 通信的基本模組，另一部分則是利用 SPI 基本模組完成對 SPI EEPROM – M2510 的讀寫操作。下面將詳細介紹這兩個部分。

一. SPI 通信基本模組， 主要包括如下#define 以及函數.

1. #define 部分說明：

a. #define Port_B

後續的條件編譯將根據這個#define 來確定用來做 SPI 通訊的 I/O PORT，如按上例的定義，程式將使用 IOB 作爲 SPI 通訊的 I/O PORT， 同樣使用者可以將例子中的 Port_B 換成 Port_A, Port_C, Port_D, Port_E, 進而更換相應的 IO Port 而不需要修改具體程式。

b. #define SerialDataOuput 1

c. #define SerialDataInput 2

d. #define SerialClock 4

e. #define ChipSelect 8

b,c,d,e 這四個#define 是用來定義具體的 SPI 的四條通信口線與 IO Port 中的 4 條 PIN 的對應關係，“1”代表使用某個 PORT 的第一條 I/O， 2 代表使用第二條 I/O，以此類推。 使用者可以通過更改這 4 個定義，隨意調配 I/O。
注意：使用目前的程式，這四條 I/O 必須在同一個 I/O Port 中。

2. 函數部分說明：

a. void F_Sim_Spi_Init(void)

SPI 信號初始化函數，這個函數要在所有相關函數被調用前執行一次，否則將可能產生問題。

b. void F_Sim_Spi_CS_Disable(void)

c. void F_Sim_Spi_CS_Enable(void)

a,b 兩項是用來控制 CHIP SELECT 信號的函數。

d. void F_Sim_Spi_Write_bit(unsigned char Data)

e. char F_Sim_Spi_Read_bit(void)

d,e 兩項是用來寫/讀一個 bit 資料的函數，這兩個函數使用 inline asm 的方式編寫，客戶可以參考其中 C 語言與 inline asm 之間參數的傳遞方法，當然，在本 AP 中這兩個函數由於屬於底層函數，也完全可以不用詳細研究。

f. unsigned char F_Sim_Spi_Read_8Bits(void)

g. void F_Sim_Spi_Write_8Bits(unsigned char ucdata)

d,e 兩項是用來讀/寫一個 byte 資料的函數，也是 SPI 通信模組的主要函數，用戶可以利用這兩個函數結合 SPI 器件的通信協議， 開發各種 SPI 器件的通信/控制程式。

二. 讀寫 M2510（SPI 介面 EEPROM）的#define 以及函數。

1. #define 部分說明：

a. #define RFD 0X03 //Read Flash Data

b. #define WRAM 0X86 //Write Data To Buffer(RAM)

c. #define RRAM 0X83 //Read Data From Buffer(RAM)

這個三個#define 是用來定義 M2510 的 SPI 通信協議中的命令代碼，主要起到增強程式可讀性的作用。

2. 函數部分說明：

a. void F_Sim_SPI_WriteToRAM_64Bytes(void)

b. void F_Sim_SPI_ReadFromRAM_64Bytes(void)

c. void F_Sim_SPI_ReadFromFlash_64Bytes(void) // 15.3ms @5.96MHz

這三個函數是根據 M2510 的 SPI 通信協議，結合 SPI 基本通信模組函數，編寫的用來演示如何完成具體 SPI 通信功能的範例函數，其中 a,c 兩個函數基本相同，都是讀取 64 bytes 資料的函數，不同點僅在於一個是從 M2510 的 BUFFER RAM 中讀取，而另一個是直接讀取 M2510 的 EEPROM 資料。 專案 b 中的函數功能是將 64 bytes 資料寫入 M2510 的 BUFFER RAM 中。

三. 主要函數詳細介紹

1. 函數 F_Sim_Spi_Write_bit

本函數的功能是將 1 bit 資料通過 SPI 介面寫入到相關器件中，寫入的資料由 C 語言函數參數的形式傳入， 而用戶可以參考本函數學習將 C 參數傳給 inline asm 的方法，具體方法請參考函數開始部分，將 C 參數位址傳給間接定址寄存器 HL。

```
void F_Sim_Spi_Write_bit(unsigned char Data)
{
   asm("shlx"  );
   asm("setdat %o",Data);                          //load Data's address to HL index register

   asm("ori*   %b,%b",PortB,DI_HIGH);              //set output data=1

   asm("lda#   @hl");                              //HL index register +1
   asm("lda    @hl");                              //load output data (the most significant bit)
   asm("jb3    L_LeftShiftOneBit");                //judge output data is 0 or not
   asm("andi*  %b,%b",PortB,DI_LOW);               //set output data=0
   asm("L_LeftShiftOneBit:");
   asm(opx     ,PortB);                            //write data to Serial Data Input pin

   asm("ori*   %b,%b",PortB,CLK_HIGH);
   asm(opx     ,PortB);                            //diriving serial clock to high
   asm("andi*  %b,%b",PortB,CLK_LOW);
   asm(opx     ,PortB);                            //diriving serial clock to low
 }
```

2. 函數 F_Sim_Spi_Read_bit

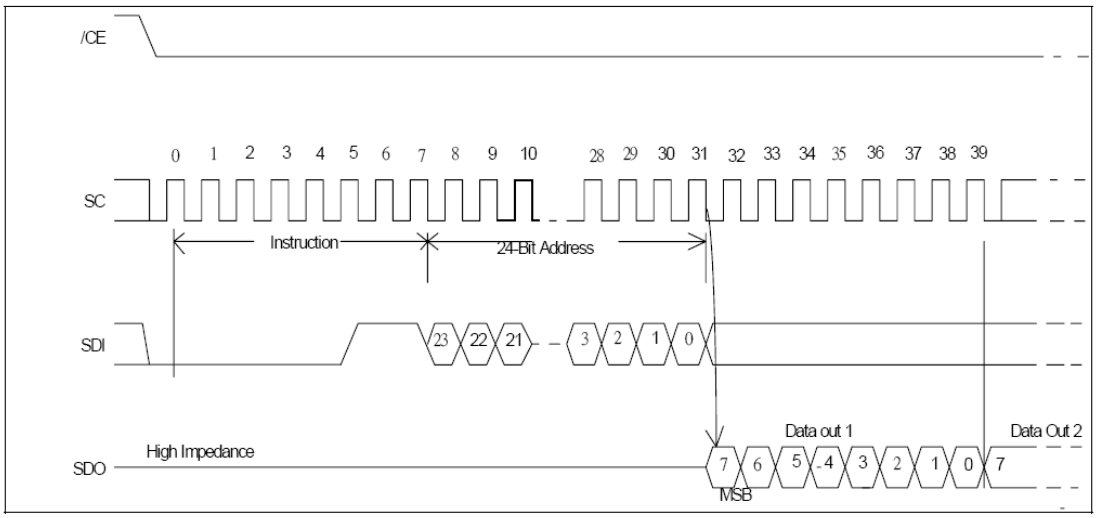
本函數的功能是通過 SPI 讀取 1 bit 的資料並將資料返回給函數的調用者，用戶需要注意的是 op1 變數， 這是 TM89 C run time library 使用的變數，其位址是 0x50， 這個變數主要用在需要使用 C run time library 做函數返回值的相關操作， 本函數雖然沒有使用到 C run time library 做函數返回值，但是仍然有使用 op1, 理論上本函數使用其他由客戶定義的變數也是可以完成這一功能的。

```
char F_Sim_Spi_Read_bit(void)
 {
   asm("andi*  %b,%b",PortB,CLK_LOW);
   asm(opx     ,PortB);                            //diriving serial clock to low
   asm("lds    op1,0");                            //set function return value=0
   asm(ipx     ,TempReg);                          //read data from Serial Data Ouput pin
   asm("andi*  %b,%b",TempReg,DO_HIGH);            //judge the input data is 0 or not
   asm("jz     InputDataIsZero");
   asm("lds    op1,1");                            //set function return value=1
   asm("InputDataIsZero:");

   asm("ori*   %b,%b",PortB,CLK_HIGH);
   asm(opx     , PortB);                           //diriving serial clock to high
   asm("andi*  %b,%b",PortB,CLK_LOW);
   asm(opx     , PortB);                           //diriving serial clock to low

   asm("lds    op1+1,0");
   asm("shlx"  );
   asm("setdat op1");                              //return value(address) in index register HL
 }
```

3. 函數 F_Sim_SPI_ReadFromFlash_64Bytes

要瞭解本函數的功能， 請先參考下面 Figure1 中的 M2510 相關時序圖。函數工作的具體步驟就是按照這個時序，調用 SPI 基本模組提供的各個函數，最終完成讀 64 bytes 的功能。 請注意在時序圖中缺少/CE（CHIP SELECT）信號返回高電平的部分，而一般而言，SPI 通信中的任何命令都是以/CE（CHIP SELECT）信號變爲低電平做爲開始，返回高電平作爲命令結束。

Figure 1 Read Flash Data (RFD) Sequence




```
void F_Sim_SPI_ReadFromFlash_64Bytes(void)
{
char c_count;
char *p;
    F_Sim_Spi_CS_Enable();          //Driving /CE low to select the device
    F_Sim_Spi_Write_8Bits(RFD);     // send the "Read Flash Data" instruction to M2510
    F_Sim_Spi_Write_8Bits(00);      // set start address for read
    F_Sim_Spi_Write_8Bits(00);      // set start address for read
    F_Sim_Spi_Write_8Bits(00);      // set start address for read
    c_count=0;
    p = Buffer;                     // Buffer is a char type array for data storage
    while(1)                        // repeat the read one byes funciton 64 times
     {
     *p=F_Sim_Spi_Read_8Bits();
     ++c_count;
     if (c_count==64) break;
     ++p;
     }
    F_Sim_Spi_CS_Disable();         //terminate RFD instruction by Driving /CE to high

}
```