CH365 利用软件模拟 SPI 和 MicroWire 接口

(版本 1.1)

1. 实现过程概述

为了验证模拟的 SPI 或 MicroWire 同步串行接口，本例中将 CH365 作为主控端，93C46 作为被控端(因为 93C46 时序与 SPI 或 MicroWire 类似)。基本原理是将每个字节分成 8 位数据在 CH365 的 A13 的同步脉冲下一位一位顺序传送，而不需要增加任何额外硬件成本。

本例使用 CH365 的 A14-A12 作为输出，D6 作为输入。

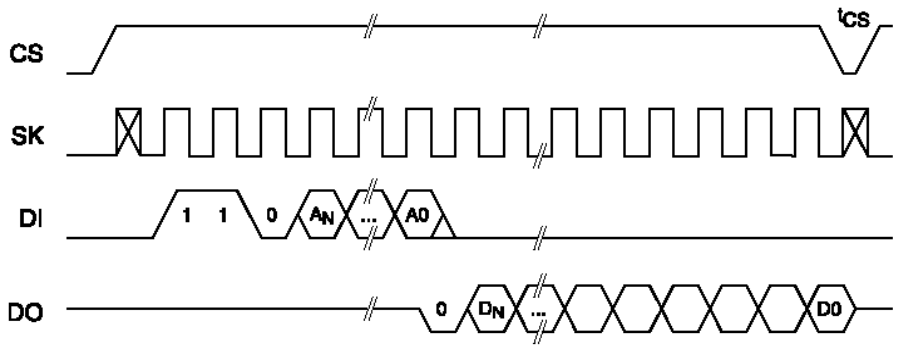
CH365 的 A14 作为片选控制信号与 93C46 的 CS 相连接。如果 A14 置为高电平，则选中 93C46；如果 A14 置为低电平，则未选中 93C46。

CH365 的 A13 作为发送脉冲信号与 93C46 的 SK 相连接。根据 A13 提供的同步脉冲信号决定 CH365 与 93C46 的同步串行通信。

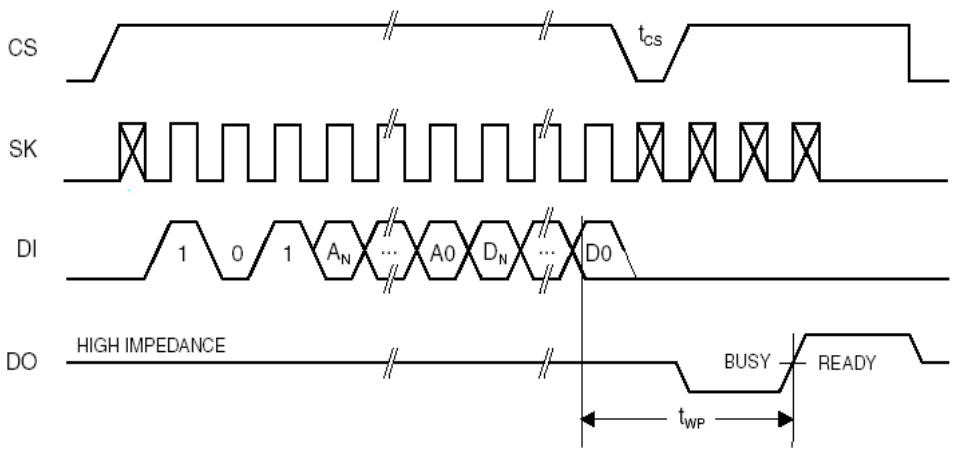
CH365 的 A12 作为输出口与 93C46 的 DI 相连接，依次输出数据。

CH365 的 D6 作为输入口与 93C46 的 DO 相连接，依次接收数据。

读时序图



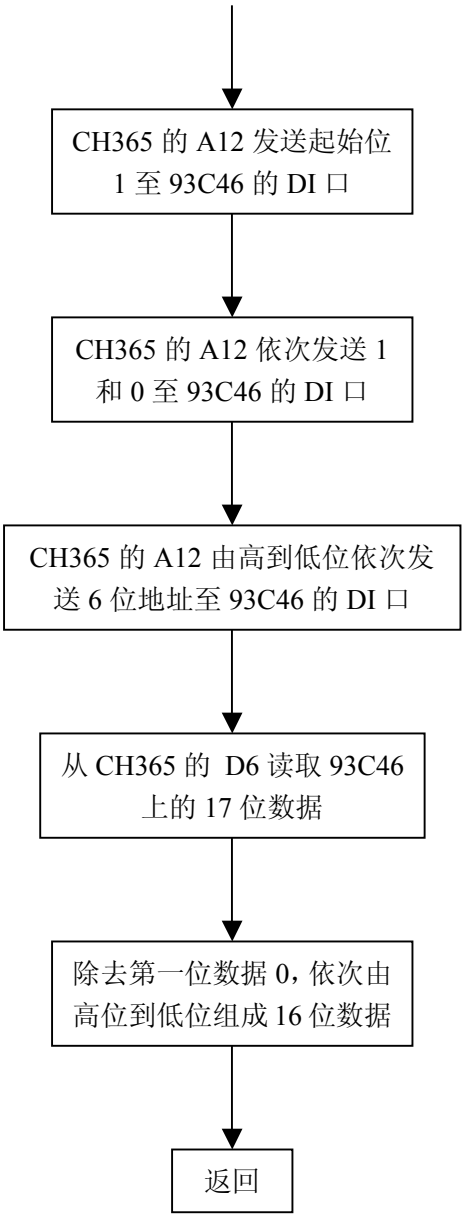
写时序图



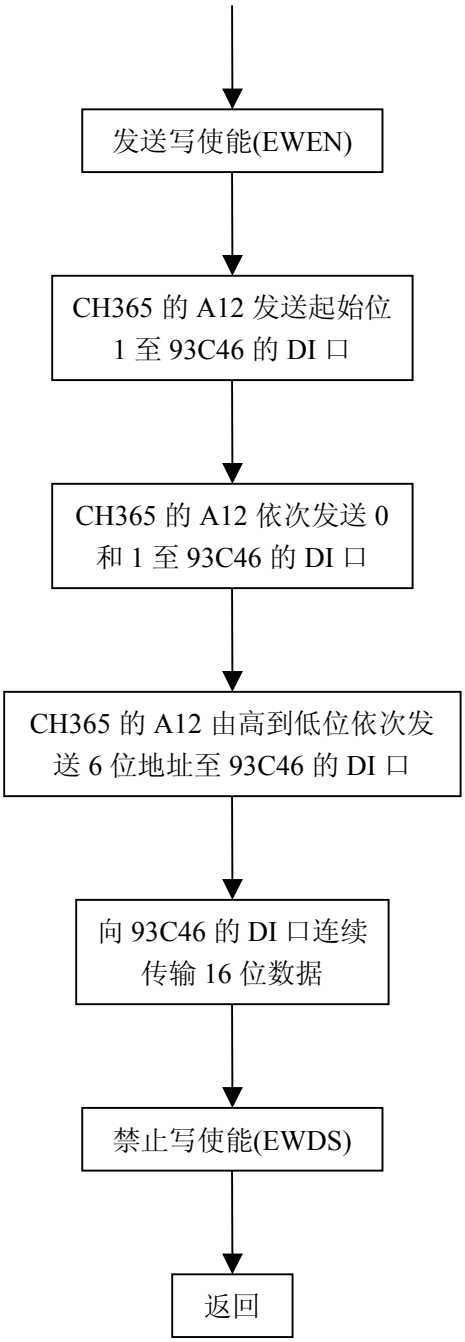
)

2. 流程图

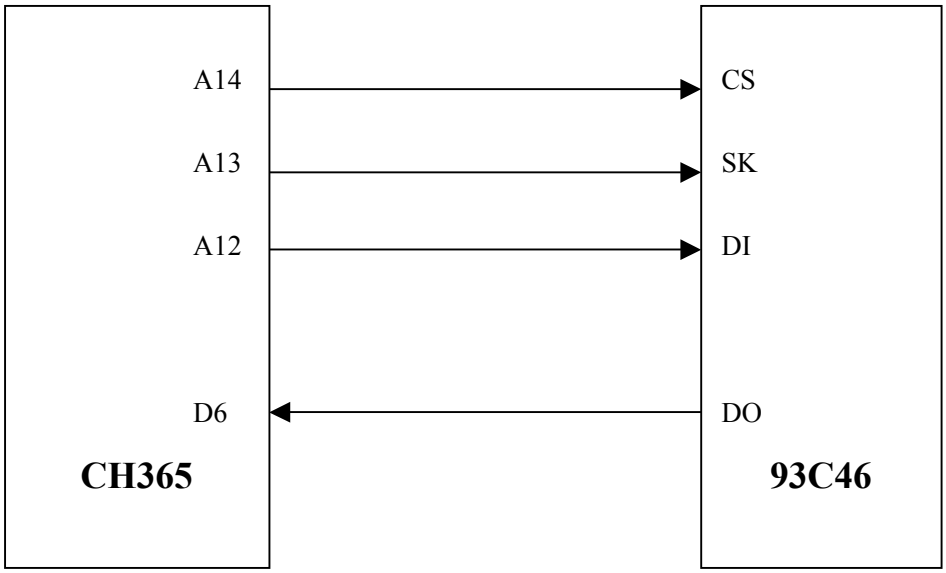
读子程序流程图



写子程序流程图



CH365 与 93C46 的接线图



3. 源程序

```
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include "CH365DOS.C"                    /*the Lib of the  CH365          */
#define BIT_CS          0x4000           /*the mask of the CS<-A14        */
#define BIT_SK          0x2000           /*the mask of the SK<-A13        */
#define BIT_DI          0x1000           /*the mask of the DI<-A12        */
#define IO_OFS          0x001F           /*I/O    Address Offset          */
#define DELAY_COUNT     20               /*Delay Adjust                   */
#define BIT_DO          0x0040           /*the mask of the DO->D6         */
#define CMD_READ        0x80             /*Read Command 10XXXXXXb         */
#define CMD_WRITE       0x40             /*Write Command 01XXXXXXb        */
#define CMD_EWEN        0x30             /*EW enable Command 0011XXXXb    */
#define CMD_ERASE       0xc0             /*ERASE Command 1100XXXXb        */
#define CMD_ERAL        0x20             /*ER ALL Command 0010XXXXb       */
#define CMD_WRAL        0x10             /*WR ALL Command 0010XXXXb       */
#define CMD_EWDS        0x00             /*EW Disable Command 0000XXXXb   */
mCH365_IO_REG           * m_IOBase;      /*i/o  base address              */
unsigned short          CH365_Addr_Swap; /*The Staus of the A15-A0         */
void    CH365_SPI_Delay();           /*The Delay Function >10ms          */
```

```
void    CH365_SPI_Clock();            /*Send clock */
void    CH365_SPI_SendCmd(unsigned char iAddr,unsigned char iCmd);
/*Send the Addr|OP. to the DI       */
void    CH365_SPI_EWEN();             /*Enable Write or Erase op.         */
void    CH365_SPI_EWDS();             /*Disable Write or Erase OP.        */
unsigned short    CH365_SPI_Read(unsigned char  iAddr);
unsigned short    CH365_SPI_Write(unsigned short iData,unsigned char iAddr);
unsigned short    CH365_SPI_ERASE(unsigned char iAddr);
unsigned short    CH365_SPI_ERASE_ALL();
unsigned short    CH365_SPI_Write_ALL(unsigned short iData);

/*************************************************************/
/************The Define of The Function Above**************/
/*************************************************************/

void  CH365_SPI_Delay()                /*The Delay times  u can adjust the upper
limits of the DELAY_COUNT*1us*/
{
    unsigned char count=0;
    for(count=0;count<DELAY_COUNT;count++) inportb(0x70);
}
void  CH365_SPI_Clock()                /*Send clock    */
{
    outport((unsigned int) & m_IOBase->mCh365MemAddrL, CH365_Addr_Swap);
    CH365_SPI_Delay();
    outport((unsigned int) & m_IOBase->mCh365MemAddrL, (CH365_Addr_Swap|BIT_SK));
    CH365_SPI_Delay();
    outport((unsigned int) & m_IOBase->mCh365MemAddrL, CH365_Addr_Swap);
    CH365_SPI_Delay();
}

/*********************CH365_SPI_SendCmd()*******************************/
/*************** Arguments: iAddr,iCmd        ***************************/
/*****把 6 位地址(iAddr)加上 2 位 OP CODE(iCmd)组成 8 位 1 个字节后传送 ****/

void  CH365_SPI_SendCmd(unsigned char iAddr,unsigned char iCmd)
{
    unsigned char count;
    CH365_Addr_Swap=inport(m_IOBase->mCh365MemAddrL);
    CH365_Addr_Swap &= ~BIT_CS;
    CH365_Addr_Swap &= ~BIT_SK;
    outport((unsigned int) & m_IOBase->mCh365MemAddrL,CH365_Addr_Swap); /* CS=0 */
    CH365_SPI_Delay();
    CH365_Addr_Swap |= BIT_CS;
```

```
    outport((unsigned int) & m_IOBase->mCh365MemAddrL,CH365_Addr_Swap);
    CH365_SPI_Delay();
    CH365_Addr_Swap |= BIT_DI;
    CH365_SPI_Clock();                    /* start bit */
    iAddr|=iCmd;
    for (count=0;count<8;count++)         /*Send The 8-Bits Addr|OP. instruction*/
    {
        if ((iAddr>>(7-count))&1) CH365_Addr_Swap |= BIT_DI;  /* bit 1 */
        else CH365_Addr_Swap &= ~ BIT_DI; /* bit 0 */
        CH365_SPI_Clock();
    }
}
/***************CH365_SPI_READDATA()   ********************/
/************* Arguments:        ***************************/
/******* return ; m_Data               ********************/
/***从 93C46 的 DO 口连续读出 16 位数据    ***************/

unsigned short CH365_SPI_READDATA()
{
    unsigned char bit_count;
    unsigned short DataSwap;
    unsigned short m_Data=0;
    CH365_SPI_Clock();            /* Get the First Data '0' From DO*/
    for (bit_count=0;bit_count<16;bit_count++)
    {
        DataSwap=inportb((unsigned int ) & m_IOBase->mCh365IoPort[IO_OFS]);
        CH365_SPI_Clock();
        m_Data<<=1;
        if (DataSwap&BIT_DO) m_Data|=1;
    }
    CH365_Addr_Swap&=~BIT_CS;
    outport((unsigned int) & m_IOBase->mCh365MemAddrL,CH365_Addr_Swap);
    return m_Data;
}
/************* CH365_SPI_SENDDATA()       ****************/
/******** Arguments:    iData             ***************/
/***  向 93C46 的 Di 口连续传输 16 位数据     ***********/

void  CH365_SPI_SENDDATA(unsigned short iData)
{
    unsigned char    bit_count=0;
    for(bit_count=0;bit_count<16;bit_count++)
    {
        if ((iData>>(15-bit_count))&1) CH365_Addr_Swap |= BIT_DI;  /* bit 1 */
```

```
        else CH365_Addr_Swap &= ~ BIT_DI; /* bit 0 */
        CH365_SPI_Clock();
    }
    CH365_Addr_Swap&=~BIT_CS;
    outport((unsigned int) & m_IOBase->mCh365MemAddrL,CH365_Addr_Swap);
}
void  CH365_SPI_EWEN()              /*enable Write:0011XXXX*/
{
    CH365_SPI_SendCmd(0,CMD_EWEN);
    CH365_Addr_Swap&=~BIT_CS;
    outport((unsigned int) & m_IOBase->mCh365MemAddrL,CH365_Addr_Swap);
}
void  CH365_SPI_EWDS()              /*Disable Write:0000XXXX*/
{
    CH365_SPI_SendCmd(0,CMD_EWDS);
    CH365_Addr_Swap&=~BIT_CS;
    outport((unsigned int) & m_IOBase->mCh365MemAddrL,CH365_Addr_Swap);
}
void  CH365_SPI_EndCheck()          /*Check the EW instruction out the process*/
{
    unsigned char DataSwap;
    unsigned long count;
    outport((unsigned int) & m_IOBase->mCh365MemAddrL,CH365_Addr_Swap&~BIT_CS);
    CH365_SPI_Delay();
    CH365_SPI_Delay();
    outport((unsigned int) & m_IOBase->mCh365MemAddrL,CH365_Addr_Swap);
    for (count=0;count<2000;count++)  /* check operation completion */
    {
        CH365_SPI_Delay();
        DataSwap=inportb((unsigned int ) & m_IOBase->mCh365IoPort[IO_OFS]);
        if (DataSwap&BIT_DO) break;
    }
    CH365_Addr_Swap&=~BIT_CS;
    outport((unsigned int) & m_IOBase->mCh365MemAddrL,CH365_Addr_Swap);
}

/********CH365_SPI_Read(unsugned short m_Address)  ********/
/********Arguments: m_Address                       ********/
/********Return Value Type: unsigned short     ************/

unsigned short  CH365_SPI_Read(unsigned char  iAddr)
{
   unsigned int m_Data;
   CH365_SPI_SendCmd(iAddr,CMD_READ);   /*Send the READ and Address OP. Data. */
```

```
    m_Data=CH365_SPI_READDATA();
    return m_Data;
}


/********************CH365_SPI_Write()*********************/
/*********Arguments:iData,iAddr                     *********/
/*********Return Value Type: unsigned short(BOOL) *********/

unsigned short  CH365_SPI_Write(unsigned short iData,unsigned char iAddr)
{
    CH365_SPI_EWEN();
    CH365_SPI_Delay();
    CH365_SPI_SendCmd(iAddr,CMD_WRITE);
    CH365_SPI_SENDDATA(iData);
    CH365_SPI_EndCheck();
    CH365_SPI_EWDS();
    return (1);
}
/**********CH365_SPI_ERASE(unsigned short iAddr)   **********/
/*********Arguments: iAddr                          *********/
/*********Return Value Type: unsigned short         *********/

unsigned short  CH365_SPI_ERASE(unsigned char iAddr)
{
    CH365_SPI_EWEN();
    CH365_SPI_Delay();
    CH365_SPI_SendCmd(iAddr,CMD_ERASE);
    CH365_SPI_EndCheck();
    CH365_SPI_EWDS();
    return(1);
}
/************* CH365_SPI_ERASE_ALL()    *******************/
/*********Arguments:                   *********************/
/*********Return Value Type: unsigned short(BOOL) *********/

unsigned short  CH365_SPI_ERASE_ALL()
{
    CH365_SPI_EWEN();
    CH365_SPI_Delay();
    CH365_SPI_SendCmd(0,CMD_ERAL);
    CH365_SPI_EndCheck();
    CH365_SPI_EWDS();
    return(1);
```

```
}

/*******CH365_SPI_Write_ALL(unsigned short iData)********/
/********Arguments:   iData                          ********/
/********Return Value Type: unsigned short(BOOL) ********/

unsigned short  CH365_SPI_Write_ALL(unsigned short iData)
{
    CH365_SPI_EWEN();
    CH365_SPI_Delay();
    CH365_SPI_SendCmd(0,CMD_WRAL);
    CH365_SPI_SENDDATA(iData);
    CH365_SPI_EndCheck();
    CH365_SPI_EWDS();
    return (1);
}

/**************The Main Fram Of The Program**************/
/************************* TEST ***************************/

int      main()
{
        int  SPI_testData;
        int  SPI_testAddr;
        printf("Now Let's Start to Check The PCI Device\n");
        if(!CH365CheckDevice( ))
        {
         printf("Sorry! Can't Find The Exact Device\n");
         exit(0);
        }
        printf("The PCI Address is %d\t\n", dosCH365PciAddr);
        m_IOBase=CH365GetIoBaseAddr();
        while ( 1 )
        {
            printf("Please Enter The Address To Write 0-63 :");
            scanf("%d",&SPI_testAddr);
            printf("\n");
            if(SPI_testAddr>64) break;
            printf("Please Enter TheDataTo Write :");
            scanf("%d",&SPI_testData);
            printf("The Input Is = %X::%d\n",SPI_testData,SPI_testData);
            printf("Start Write\n");
            if(!CH365_SPI_Write(SPI_testData,SPI_testAddr))
            {
```

```
                printf("error Writing !!!!\n");
                exit(0);
            }
            printf("start Read!");
            SPI_testData=CH365_SPI_Read(SPI_testAddr);
            printf("The Test Data %d=%x\n",SPI_testAddr,SPI_testData);
        }
}

/*****************************End of The Process*****************************/
/****************************************************************************/
```