```
/********************************** (C) COPYRIGHT *******************************
* File Name          : SPI_HW.C
* Author             : WXF
* Version            : V1.0
* Date               : 2013/12/19
* Description        : CH395 芯片  CH395 芯片  硬件标准 SPI 串行连接的硬件抽象层  V1.0
*                      提供 I/O 接口子程序
*******************************************************************************
/
#include "CH395SPI.H"
#include "delay.h"
#include "CH395INC.H"


/*******************************************************************************
*
* Function Name  : CH395_Port_Init
* Description    : CH395 端口初始化
*                  由于使用 SPI 读写时序,所以进行初始化
* Input          : None
* Output         : None
* Return         : None
*******************************************************************************
/
void CH395_PORT_INIT( void )
{
    //替换自己的端口初始化函数
    spi_parameter_struct    SPI_InitStructure;


    rcu_periph_clock_enable(RCU_SPI0);
  rcu_periph_clock_enable(RCU_GPIOA);
    rcu_periph_clock_enable(RCU_GPIOB);
    rcu_periph_clock_enable(RCU_GPIOC);
    rcu_periph_clock_enable(RCU_GPIOD);

    // Configure pins: SCK, MISO and MOSI
    gpio_af_set(GPIOA, GPIO_AF_5,GPIO_PIN_5);
  gpio_mode_set(GPIOA, GPIO_MODE_AF, GPIO_PUPD_NONE,GPIO_PIN_5 );
  gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ,GPIO_PIN_5);

    gpio_af_set(GPIOB, GPIO_AF_5,GPIO_PIN_5 );
  gpio_mode_set(GPIOB, GPIO_MODE_AF, GPIO_PUPD_NONE,GPIO_PIN_5 );
  gpio_output_options_set(GPIOB, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ,GPIO_PIN_5 );
```

```
    gpio_mode_set(GPIOA, GPIO_MODE_INPUT, GPIO_PUPD_NONE,GPIO_PIN_6);
    //CS
    gpio_mode_set(GPIOA, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE,GPIO_PIN_4 );
  gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ,GPIO_PIN_4 );

    //RST 复位
    gpio_mode_set(GPIOD, GPIO_MODE_OUTPUT, GPIO_PUPD_PULLUP,GPIO_PIN_2 );
  gpio_output_options_set(GPIOD, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ,GPIO_PIN_2 );

    //INT 中断
  gpio_mode_set(GPIOC, GPIO_MODE_INPUT, GPIO_PUPD_PULLUP,    GPIO_PIN_13);

    /*CS high */
    gpio_bit_set(GPIOA,GPIO_PIN_4);

    /* SPI configuration */
    SPI_InitStructure.trans_mode = SPI_TRANSMODE_FULLDUPLEX; /* SPI 配置成两线的单向
全双工通信 */
    SPI_InitStructure.device_mode = SPI_MASTER;                    /* SPI 主机 */
    SPI_InitStructure.frame_size = SPI_FRAMESIZE_8BIT;             /* SPI8 位数据格式
传输 */
    SPI_InitStructure.clock_polarity_phase = SPI_CK_PL_LOW_PH_1EDGE;
          /* 时钟低时活动 */
    SPI_InitStructure.nss = SPI_NSS_SOFT;                          /* 内部 NSS 信号由 SSI
控制 */
    SPI_InitStructure.prescale = SPI_PSC_32; /* 波特率预分频数为 4 */
    SPI_InitStructure.endian = SPI_ENDIAN_MSB;                     /* 传输时高位在前 */

    spi_init(SPI0, &SPI_InitStructure );
//  spi_crc_polynomial_set(SPI0,7);
    /* Enable SPI  */
    spi_enable(SPI0);

}

/*******************************************************************************
*
* Function Name  : Spi395Exchange
* Description    : 硬件 SPI 输出且输入 8 个位数据
* Input          : d---将要送入到 CH395 的数据
* Output         : None
* Return         : SPI 接收的数据
*******************************************************************************
```

```
/
uint8_t Spi395Exchange( uint8_t d )
{
        uint8_t temp = 0;
        gpio_bit_reset(GPIOA,GPIO_PIN_4);
        /* Loop while DR register in not emplty */
        while(RESET == spi_i2s_flag_get(SPI0,SPI_FLAG_TBE));

    /* send byte through the SPI0 peripheral */
    spi_i2s_data_transmit(SPI0,d);

    /* wait to receive a byte */
    while(RESET == spi_i2s_flag_get(SPI0,SPI_FLAG_RBNE));

    /* return the byte read from the SPI bus */
    temp=spi_i2s_data_receive(SPI0);
        gpio_bit_set(GPIOA,GPIO_PIN_4);
        return temp;
}

/*******************************************************************************
*
* Function Name  : xWriteCH395Cmd
* Description    : 向 CH395 写命令
* Input          : mCmd---将要写入 CH395 的命令码
* Output         : None
* Return         : None
*******************************************************************************
/
void xWriteCH395Cmd( uint8_t mCmd )
{
    gpio_bit_set(GPIOA,GPIO_PIN_4);
    /* 对于双向 I/O 引脚模拟 SPI 接口,那么必须确保已经设置 SPI_SCS,SPI_SCK,SPI_SDI 为输
出方向,SPI_SDO 为输入方向 */
    gpio_bit_reset(GPIOA,GPIO_PIN_4);                                   /* SPI
片选有效 */

    /* 发送命令码 */
    Spi395Exchange(mCmd);                                               /* 发出命令码 */
    delay_us( 2 );                                                      /* 延时 1.5uS 确保读
写周期大于 1.5uS */
}

/*******************************************************************************
```

```
*
* Function Name   : xWriteCH395Data
* Description     : 向 CH395 写数据
* Input           : mData---将要写入 CH395 的数据
* Output          : None
* Return          : None
*******************************************************************************
/
void xWriteCH395Data( uint8_t mData )
{
    Spi395Exchange(mData);                                          /* 发送数据 */
}

/*******************************************************************************
*
* Function Name   : xReadCH395Data
* Description     : 从 CH395 读数据
* Input           : None
* Output          : None
* Return          : 返回读取的数据
*******************************************************************************
/
uint8_t xReadCH395Data( void )
{
    return(Spi395Exchange(0xFF));
}


/*******************************************************************************
*
* Function Name   : Query395Interrupt
* Description     : 查询 CH395 中断(INT#低电平)
* Input           : None
* Output          : None
* Return          : 返回中断状态
*******************************************************************************
/
uint8_t Query395Interrupt( void )
{
    return( gpio_input_bit_get(GPIOC,GPIO_PIN_13) ? RESET   : SET   );
}

/*******************************************************************************
*
```

```
* Function Name    : CH395_RST
* Description      : 复位 CH395
* Input            : None
* Output           : None
* Return           : 返回中断状态
*******************************************************************************
/
void CH395_RST( void )
{
    gpio_bit_reset(GPIOD,GPIO_PIN_2);
    delay_ms(250);
    gpio_bit_set(GPIOD,GPIO_PIN_2);
    delay_ms(250);

}
```