```
//VK3222_VK3224_VK3232_VK3233_VK3234_VK3266_VK3366_VK3368_VK3362
                        SPI_TO_UART 驱动 C 语言代码
/*
SPI 接口宽工作电压 2 通道 16 级 FIFO 的 UART VK3222 SSOP16
http://www.fosvos.com/datasheet/VKFV/VK3222_DS_CN_FV_V014.pdf
SPI 接口宽工作电压 4 通道 16 级 FIFO 的 UART VK3224 SOP20
http://www.fosvos.com/datasheet/VKFV/VK3224_DS_CN_FV_V012.pdf
SPI/UART 接口宽工作电压 2 通道 16 级 FIFO 的 UART VK3232 SOP24
http://www.fosvos.com/datasheet/VKFV/VK3232_DS_CN_FV_V012.pdf
SPI/UART 接口宽工作电压 3 通道 16 级 FIFO 的 UART VK3233 SOP28
http://www.fosvos.com/datasheet/VKFV/VK3233_DS_CN_FV_V011.pdf
SPI/UART 接口宽工作电压 4 通道 16 级 FIFO 的 UART VK3234 SOP28
http://www.fosvos.com/datasheet/VKFV/VK3234_DS_CN_FV_V012.pdf
SPI/UART/8 位并行总线接口宽工作电压 4 通道 16 级 FIFO 的 UART VK3266 QFP44
http://www.fosvos.com/datasheet/VKFV/VK3266_DS_CN_FV_V013.pdf
SPI/ UART /8 位并行总线接口 1.8V QFN 封装 4 通道 16 级 FIFO 的 UART VK3268 QFN32
http://www.fosvos.com/datasheet/VKFV/VK3268_DS_CN_FV_V01.pdf
SPI/I IC/ UART /8 位并行总线接口宽工作电压 2 通道 16 级 FIFO 的 UART VK3362 SOP28
http://www.fosvos.com/datasheet/VKFV/VK3362_DS_CN_FV_V01.pdf
SPI/I IC/ UART /8 位并行总线接口宽工作电压 4 通道 16 级 FIFO 的 UART VK3366 QFP44
http://www.fosvos.com/datasheet/VKFV/VK3366_DS_CN_FV_V01.pdf
SPI/IIC/UART/8 位并行总线接口 1.8V QFN 封装 4 通道 16 级 FIFO 的 UART VK3368 QFN32
http://www.fosvos.com/datasheet/VKFV/VK3368_DS_CN_FV_V012.pdf
*/
////////////////////////////////////////////////////////////////////////////////////////////////
//
//          51 单片机 采用 SPI_TO_UART
//     上海福跃电子提供技术销售服务     www.fosvos.com
//     TEL: 021-58998693\58994470-15      QQ:42915158
//       main.c
//
////////////////////////////////////////////////////////////////////////////////////////////////
#include "def.h"
#include "vk33.h"
#include <stdio.h>

//在各种方式中，main.c 函数和 vkxx.h 是通用的 def.h 在 spi 和 parallel 方式是通用的 uart
//的多了两个批量读写 fifo 的函数；底层函数不同

void delay (unsigned int delaydata)
{
      unsigned int tempdata=delaydata;
      while(tempdata)
      {
```

```
    tempdata=tempdata-1;
    }
}


void Init_led(void)
{
    unsigned int tempdata=0x0ffff;

    CH4_LED = 0;
    CH3_LED = 0;
    CH2_LED = 0;
    CH1_LED = 0;
    PAR_LED = 0;
    SPI_LED = 0;
    UART_LED = 0;
    IIC_LED   =  0;
    while(tempdata)
    {
    tempdata=tempdata-1;
    }
    CH4_LED = 1;
    CH3_LED = 1;
    CH2_LED = 1;
    CH1_LED = 1;
    PAR_LED = 1;
    SPI_LED = 1;
    UART_LED = 1;
    IIC_LED   =  1;
    tempdata=0x0ffff;
    while(tempdata)
    {
    tempdata=tempdata-1;
    }
    CH4_LED = 0;
    CH3_LED = 0;
    CH2_LED = 0;
    CH1_LED = 0;
    PAR_LED = 0;
    SPI_LED = 0;
    UART_LED = 0;
    IIC_LED   =  0;
    tempdata=0x0ffff;
    while(tempdata)
```

```
    {
    tempdata=tempdata-1;
    }
    CH4_LED = 1;
    CH3_LED = 1;
    CH2_LED = 1;
    CH1_LED = 1;
    PAR_LED = 1;
    SPI_LED = 1;
    UART_LED = 1;
    IIC_LED   =  1;
    tempdata=0x0ffff;
    while(tempdata)
    {
    tempdata=tempdata-1;
    }
    CH4_LED = 0;
    CH3_LED = 0;
    CH2_LED = 0;
    CH1_LED = 0;
    PAR_LED = 0;
    SPI_LED = 0;
    UART_LED = 0;
    IIC_LED   =  0;
    tempdata=0x0ffff;
    while(tempdata)
    {
    tempdata=tempdata-1;
    }
}


main()
{

    static unsigned char dat1,dat2,dat3,dat4,m,n;
    unsigned int tmp=0xfff;
    unsigned char err_fla=0,err_flb=0,err_flc=0,err_fld=0;
    n=0;
    m=0;
    //初试化 MCU
    config();
    //初试化 LED
    Init_led();
```

```
//初始化 VK；
config_vkxx();
//主 SPI 接口测试;亮起 SPI_LED
CH4_LED = 1;
CH3_LED = 1;
CH2_LED = 1;
CH1_LED = 1;
PAR_LED = 1;
SPI_LED = 0;
UART_LED = 1;
IIC_LED   = 1;

while(1)
{
//写出数据分别到 4 个子 UART
      write_reg(1,SFDR,m);
      write_reg(2,SFDR,m);
      write_reg(3,SFDR,m);
      write_reg(4,SFDR,m);
//延迟后读取 FIFO 数据
      delay(0x8ff);
      delay(0x8ff);
      dat1=read_reg(1,SFDR);
      dat2=read_reg(2,SFDR);
      dat3=read_reg(3,SFDR);
      dat4=read_reg(4,SFDR);
      //delay(0x8ff);
      //delay(0x8ff);
//比较读回的数据是否为发出的数据后设置 LED 的状态。
    if(dat1==m)
       CH1_LED =0;
    else
       err_fla =1;
    if(dat2==m)
       CH2_LED =0;
    else
       err_flb =1;
    if(dat3==m)
       CH3_LED =0;
    else
       err_flc =1;
    if(dat4==m)
       CH4_LED =0;
    else
```

```
            err_fld =1;

      //*********************************************
      //如果有一次不等将熄灭灯

            if(err_fla==1)
              CH1_LED =1;
            if(err_flb==1)
              CH2_LED =1;
            if(err_flc==1)
              CH3_LED =1;
            if(err_fld==1)
              CH4_LED =1;
        m=m+1;
        if(m==255)
           m=0;
    }
}

////////////////////////////////////////////////////////////////////////////////////////////////
//
//          51 单片机 采用 SPI_TO_UART
//     上海福跃电子提供技术销售服务     www.fosvos.com
//     TEL: 021-58998693\58994470-15      QQ:42915158
//       SPI.c
//
////////////////////////////////////////////////////////////////////////////////////////////////
#include "stc51reg.h"
#include <intrins.h>
//************************************************************
void config()
{
      WDT_CONTR = 0x00;      // Watchdog Timer Control Register

}
//初始化单片机，
//************************************************************

//************************************************************
unsigned char send(unsigned char _data1,unsigned char _data2)
{     unsigned char i;
      clk=0;
      ness=0;
      i=0;
```

```
    if(_data1&0x80)
          mosi=1;
     else mosi=0;

     clk=1;

//   if(miso==1)
//         i[0]=i[0]+0x80;
     clk=0;


    if(_data1&0x40)
     {
          mosi=1;
     }
     else mosi=0;

     clk=1;



//   if(miso==1)
//         i[0]=i[0]+0x40;
      clk=0;

    if(_data1&0x20)
     {
          mosi=1;
     }
     else mosi=0;

     clk=1;



//   if(miso==1)
//         i[0]=i[0]+0x20;
     clk=0;

    if(_data1&0x10)
     {
          mosi=1;
```

```
        }
        else mosi=0;

        clk=1;



//      if(miso==1)
//              i[0]=i[0]+0x10;
        clk=0;

      if(_data1&0x08)
        {
              mosi=1;
        }
        else mosi=0;

        clk=1;



//      if(miso==1)
//              i[0]=i[0]+0x08;
        clk=0;

      if(_data1 & 0x04)
        {
              mosi=1;
        }
        else mosi=0;

        clk=1;



//      if(miso==1)
//              i[0]=i[0]+0x04;
        clk=0;

      if(_data1&0x02)
        {
              mosi=1;
        }
        else mosi=0;
```

```
        clk=1;


//      if(miso==1)
//              i[0]=i[0]+0x02;
        clk=0;

       if(_data1&0x01)
        {
              mosi=1;
        }
        else mosi=0;

        clk=1;


//      if(miso==1)
//              i[0]=i[0]+0x01;
        clk=0;


       if(_data2&0x80)
        {
              mosi=1;
        }

        else mosi=0;

        clk=1;


        if(miso)
              i=i+128;
        clk=0;

       if(_data2&0x40)
        {
              mosi=1;
        }
        else mosi=0;


        clk=1;
```

```
 if(miso)
       i=i+64;
 clk=0;


if(_data2&0x20)
       mosi=1;
 else mosi=0;

 clk=1;

 if(miso)
       i=i+32;

 clk=0;


if(_data2&0x10)
       mosi=1;
 else mosi=0;


 clk=1;

 if(miso)
       i=i+16;
 clk=0;


if(_data2&0x08)
       mosi=1;
 else mosi=0;


 clk=1;

 if(miso)
       i=i+8;
 clk=0;

if(_data2&0x04)
       mosi=1;
 else mosi=0;
```

```
    clk=1;

    if(miso)
        i=i+4;
    clk=0;

  if(_data2&0x02)
        mosi=1;
    else mosi=0;


    clk=1;

    if(miso)
        i=i+2;
    clk=0;

  if(_data2&0x01)
        mosi=1;
    else mosi=0;

    clk=1;

    if(miso)
        i=i+1;
    clk=0;
    mosi=0;
    ness=1;

    return i;
}
//模拟 SPI 时序，data1,data2 是写入的两个数据，这里仅返回第二个数据
//*****************************************************************************

//*****************************************************************************
void write_reg(unsigned char port,unsigned char reg,unsigned char dat)
{
    send(0x60+port*32+reg*2,dat);
}
//写寄存器，port 为子串口的路数,reg 为寄存器的地址，dat 为写入寄存器的数据
//*****************************************************************************
```

```
//*****************************************************************************
unsigned char read_reg(unsigned char port,unsigned char reg)
{
    return send(((port-1)<<5)+(reg<<1),0x00);
}
//读寄存器，珼 ort 为子串口的路数,reg 为寄存器的地址，返回寄存器的值
//*****************************************************************************

////////////////////////////////////////////////////////////////////////////////////////////////
//
//        51 单片机 采用 SPI 转 UART      VK33XX
//    上海福跃电子提供技术销售服务    www.fosvos.com
//    TEL: 021-58998693\58994470-15     QQ:42915158
//      VK33.c
//
////////////////////////////////////////////////////////////////////////////////////////////////
#include "def.h"
//*****************************************************************************
*******
void config_vkxx()
{
//*************************************************************
    write_reg(1,SCTLR,0X38);
    write_reg(2,SCTLR,0X38);
    write_reg(3,SCTLR,0X38);
    write_reg(4,SCTLR,0X38);
//使能子串口，设置子串口的波特率，具体参看数据手册中，子串口波特率设置
//*************************************************************

//*************************************************************
    write_reg(1,SFOCR,0XFF);
    write_reg(2,SFOCR,0XFF);
    write_reg(3,SFOCR,0XFF);
    write_reg(4,SFOCR,0XFF);


    write_reg(1,SFOCR,0XFC);
    write_reg(2,SFOCR,0XFC);
    write_reg(3,SFOCR,0XFC);
    write_reg(4,SFOCR,0XFC);
//清空发送接收 FIFO 中的数据，使能发送接收 FIFO
//*************************************************************

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

```
    while(read_reg(1,SFSR))
        read_reg(1,SFDR);
    while(read_reg(2,SFSR))
        read_reg(2,SFDR);
    while(read_reg(3,SFSR))
        read_reg(3,SFDR);
    while(read_reg(4,SFSR))
        read_reg(4,SFDR);
//查看发送接收 FIFO 中是否有数据，如果有则把 FIFO 中的数据读出来，
//使发送接收 FIFO 中的数据为 0
//*************************************************************
}
//*****************************************************************************
****

//*********************************************************
void stop_bit(port)
{
    write_reg(port-1,SCONR,0X48);
}
//需要说明的是：有奇偶校验的时候，数据位必须设置为 9 位;这里是偶校验
//这里还可以设置该路子串口的强制校验模式
//*************************************************************

//*************************************************************
void net_add(port)
{   unsigned char reg;
    reg=read_reg(port-1,SCONR);
    write_reg(port-1,SCONR,reg|0x03);
}
//设置网络地址可见设置，网络地址自动识别，网络地址是否
//写入 FIFO 的设置
//*********************************************************

//*********************************************************
void address(unsigned char port,unsigned char add)
{
    write_reg(port-1,SADR,add);
}
//485 模式下的网络地址
//*********************************************************
//*********************************************************
void mode(port)
{
```

```
    write_reg(port-1,SCTLR,0X38);
}
//子串口波特率设置为上电默认值，使能该子串口，设置模?
//普通 232 模式，非 IR 模式。这里可以设置为 485 模式，红外 232 模式
//还可以设置该路子串口是否接收广播数据
//*****************************************************

//*****************************************************
void aoto_flow(port)
{
    write_reg(port-1,SFWCR,0XCE);
}
//设置自动硬件流量控制
//若要改为自动软件流量控制将 0XCE 改为 0X09，写 GXOFF 和 GXON 寄存器即可
//这里还可以设置暂停发送数据和继续发送数据的触发点
//*****************************************************

//*****************************************************
void idle()
{
    write_reg(1,GCR,0X40);
}
//进入低功耗模式；把 4 改为零即可进入正常模式
//*****************************************************

//*****************************************************
void broadcast()
{
    write_reg(1,GCR,0X80);
}
//使能广播模式，把 8 改为 0 即可进入正常模式
//*****************************************************

//*****************************************************
void m_uart()
{
    write_reg(1,GUCR,0X30);
}
//主串口控制，波特率设置为默认值，无校验，1 位停止位
//*****************************************************

//*****************************************************
void gx_off()
{
```

```
    write_reg(1,GXOFF,0X48);
}
//gxoff 的值设置为 0x48
//*********************************************************

//*********************************************************
void gx_on()
{
    write_reg(1,GX0N,0X58);
}
//gxoff 的值设置为 0x58
//*********************************************************

////////////////////////////////////////////////////////////////////////////////////////////////
//
//          51 单片机 采用 SPI 转 UART       VK33XX
//     上海福跃电子提供技术销售服务     www.fosvos.com
//     TEL: 021-58998693\58994470-15     QQ:42915158
//       VK33.h
//
////////////////////////////////////////////////////////////////////////////////////////////////
extern void config_vkxx();
extern void stop_bit(port);
extern void mode(port);
extern void aoto_flow(port);
extern void idle();
extern void broadcast();
extern void m_uart();
extern void gx_off();
extern void gx_on();
extern void net_add(port);
extern void address(unsigned char port,unsigned char add);

////////////////////////////////////////////////////////////////////////////////////////////////
//
//          51 单片机 采用 PAR8 转 UART       VK33XX
//     上海福跃电子提供技术销售服务     www.fosvos.com
//     TEL: 021-58998693\58994470-15     QQ:42915158
//       def.h
//
////////////////////////////////////////////////////////////////////////////////////////////////
#include "stc51reg.h"
#define  GCR          0X01
#define  GUCR         0X02
```

```
#define  GIR          0X03
#define  GXOFF        0X04
#define  GX0N         0X05
#define  SCTLR        0X06
#define  SCONR        0X07
#define  SFWCR        0X08
#define  SFOCR        0X09
#define  SADR         0X0A
#define  SIER         0X0B
#define  SIFR         0X0C
#define  SSR          0X0D
#define  SFSR         0X0E
#define  SFDR         0X0F
sbit reset=P0^1;

extern void config();
extern unsigned char send(unsigned char _data1,unsigned char _data2);
extern void write_reg(unsigned char port,unsigned char reg,unsigned char dat);
extern unsigned char read_reg(unsigned char port,unsigned char reg);

////////////////////////////////////////////////////////////////////////////////////////////////
//
//          51 单片机 采用 SPI 转 UART       VK33XX
//     上海福跃电子提供技术销售服务     www.fosvos.com
//     TEL: 021-58998693\58994470-15     QQ:42915158
//       stc51reg.h
//
////////////////////////////////////////////////////////////////////////////////////////////////
/* After is STC additional SFR */

/* sfr   AUXR   = 0x8e; */
/* sfr   AUXR1 = 0xa2; */
/* sfr   IPH    = 0xb7; */

sfr   P4   = 0xe8;
sbit P43 = P4^3;
sbit P42 = P4^2;
sbit P41 = P4^1;
sbit P40 = P4^0;

sfr   XICON = 0xc0;

sfr   WDT_CONTR = 0xe1;
```

```
sfr ISP_DATA  = 0xe2;
sfr ISP_ADDRH = 0xe3;
sfr ISP_ADDRL = 0xe4;
sfr ISP_CMD   = 0xe5;
sfr ISP_TRIG  = 0xe6;
sfr ISP_CONTR = 0xe7;

/* Above is STC additional SFR */

/*--------------------------------------------------------------------------
REG51F.H

Header file for 8xC31/51, 80C51Fx, 80C51Rx+
Copyright (c) 1988-1999 Keil Elektronik GmbH and Keil Software, Inc.
All rights reserved.

Modification according to DataSheet from April 1999
  - SFR's AUXR and AUXR1 added for 80C51Rx+ derivatives
--------------------------------------------------------------------------*/

/*   BYTE  Registers   */
sfr P0    = 0x80;
sfr P1    = 0x90;
sfr P2    = 0xA0;
sfr P3    = 0xB0;
sfr PSW   = 0xD0;
sfr ACC   = 0xE0;
sfr B     = 0xF0;
sfr SP    = 0x81;
sfr DPL   = 0x82;
sfr DPH   = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0   = 0x8A;
sfr TL1   = 0x8B;
sfr TH0   = 0x8C;
sfr TH1   = 0x8D;
sfr IE    = 0xA8;
sfr IP    = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;

/*   80C51Fx/Rx  Extensions   */
```

```
sfr AUXR    = 0x8E;
sfr AUXR1   = 0xA2;
sfr SADDR   = 0xA9;
sfr IPH     = 0xB7;
sfr SADEN   = 0xB9;
sfr T2CON   = 0xC8;
sfr T2MOD   = 0xC9;
sfr RCAP2L = 0xCA;
sfr RCAP2H = 0xCB;
sfr TL2     = 0xCC;
sfr TH2     = 0xCD;

/* PCA SFR
sfr CCON    = 0xD8;
sfr CMOD    = 0xD9;
sfr CCAPM0 = 0xDA;
sfr CCAPM1 = 0xDB;
sfr CCAPM2 = 0xDC;
sfr CCAPM3 = 0xDD;
sfr CCAPM4 = 0xDE;
sfr CL      = 0xE9;
sfr CCAP0L = 0xEA;
sfr CCAP1L = 0xEB;
sfr CCAP2L = 0xEC;
sfr CCAP3L = 0xED;
sfr CCAP4L = 0xEE;
sfr CH      = 0xF9;
sfr CCAP0H = 0xFA;
sfr CCAP1H = 0xFB;
sfr CCAP2H = 0xFC;
sfr CCAP3H = 0xFD;
sfr CCAP4H = 0xFE;
*/

/*  BIT Registers  */
/*  PSW    */
sbit CY    = PSW^7;
sbit AC    = PSW^6;
sbit F0    = PSW^5;
sbit RS1   = PSW^4;
sbit RS0   = PSW^3;
sbit OV    = PSW^2;
sbit P     = PSW^0;
```

```
/*    TCON    */
sbit  TF1   = TCON^7;
sbit  TR1   = TCON^6;
sbit  TF0   = TCON^5;
sbit  TR0   = TCON^4;
sbit  IE1   = TCON^3;
sbit  IT1   = TCON^2;
sbit  IE0   = TCON^1;
sbit  IT0   = TCON^0;

/*    IE      */
sbit  EA    = IE^7;
sbit  EC    = IE^6;
sbit  ET2   = IE^5;
sbit  ES    = IE^4;
sbit  ET1   = IE^3;
sbit  EX1   = IE^2;
sbit  ET0   = IE^1;
sbit  EX0   = IE^0;

/*    IP      */
/*    sbit  PPC   = IP^6;*/
sbit  PT2   = IP^5;
sbit  PS    = IP^4;
sbit  PT1   = IP^3;
sbit  PX1   = IP^2;
sbit  PT0   = IP^1;
sbit  PX0   = IP^0;

/*    P3    */
sbit  RD    = P3^7;
sbit  WR    = P3^6;
sbit  T1    = P3^5;
sbit  T0    = P3^4;
sbit INT1 = P3^3;
sbit INT0 = P3^2;
sbit  TXD   = P3^1;
sbit  RXD   = P3^0;

/*    SCON    */
sbit SM0  = SCON^7; // alternatively "FE"
sbit  FE    = SCON^7;
sbit  SM1   = SCON^6;
sbit  SM2   = SCON^5;
```

```
sbit REN  = SCON^4;
sbit TB8  = SCON^3;
sbit RB8  = SCON^2;
sbit TI   = SCON^1;
sbit RI   = SCON^0;

/*  P1  */
/* PCA
sbit CEX4 = P1^7;
sbit CEX3 = P1^6;
sbit CEX2 = P1^5;
sbit CEX1 = P1^4;
sbit CEX0 = P1^3;
sbit ECI  = P1^2;
*/

sbit T2EX = P1^1;
sbit T2   = P1^0;

/*  T2CON  */
sbit TF2   = T2CON^7;
sbit EXF2  = T2CON^6;
sbit RCLK  = T2CON^5;
sbit TCLK  = T2CON^4;
sbit EXEN2 = T2CON^3;
sbit TR2   = T2CON^2;
sbit C_T2  = T2CON^1;
sbit CP_RL2= T2CON^0;

/*  CCON  */
/*  PCA
sbit CF    = CCON^7;
sbit CR    = CCON^6;
sbit CCF4  = CCON^4;
sbit CCF3  = CCON^3;
sbit CCF2  = CCON^2;
sbit CCF1  = CCON^1;
sbit CCF0  = CCON^0;
*/

//*****************************
//指示 LED 定义
sbit CH4_LED   = P1^0;
sbit CH3_LED   = P1^1;
```

```
sbit CH2_LED    = P1^2;
sbit CH1_LED    = P1^3;
sbit PAR_LED    = P1^4;
sbit SPI_LED    = P1^5;
sbit UART_LED   = P1^6;
sbit IIC_LED    = P1^7;


//*****************************
//SPI 引脚定义

sbit ness=P3^7;
sbit clk =P1^7;
sbit sda=P3^1;
sbit miso=P3^0;

//IIC 引脚定义
/*
sbit clk =P3^1;
sbit sda=P3^0;
sbit IICA0=P0^1;
sbit IICA1=P0^0;
*/
//PAR 控制引脚定义
/*
sbit PRD=P3^1;
sbit PWR=P3^6;
sbit A0=P3^0;
sbit CS=P2^0;
*/
```