单片机与组态王的通信

组态王（kingView）内置了通用单片机通信模块，这样，我们自己开发的单片机仪表就可以挂接在KingView 上了。因为这样，所以对这个东西有了些兴趣，做了些研究。

（1）研究环境

组态王 6.53，免费下载，当然有使用限制，不过用于研究是没有问题的。下载地址： http://www.kingview.com/download/index.aspx

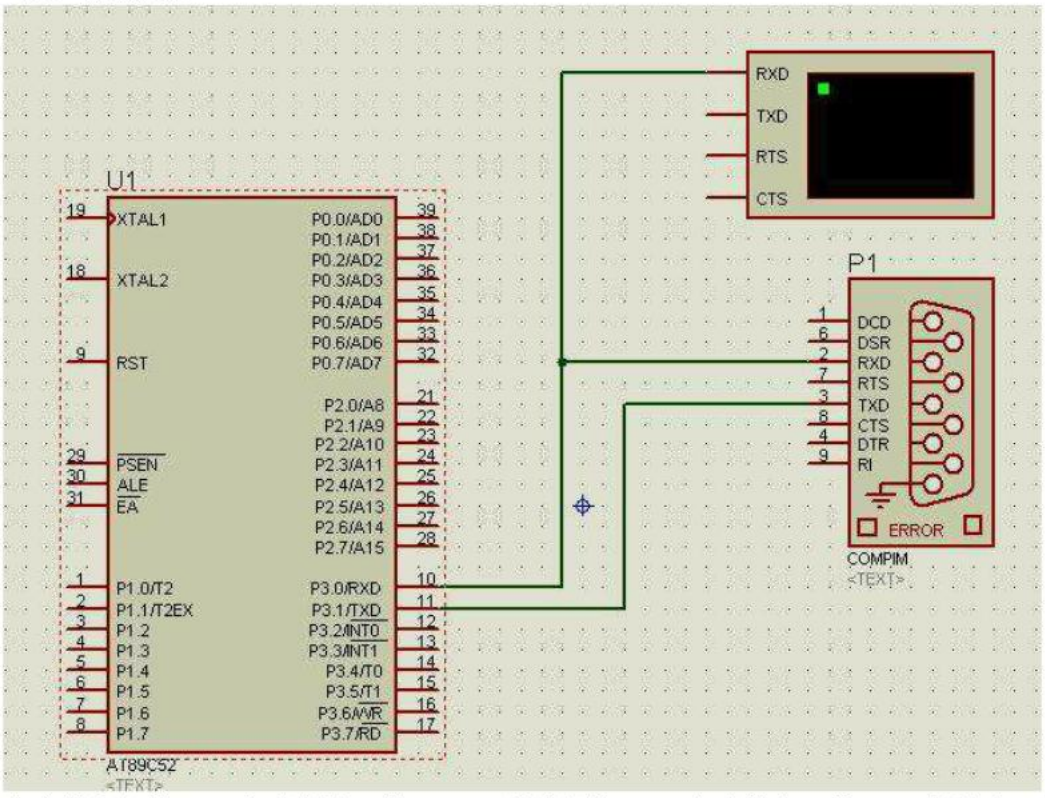
Keil 软件，Porteus，这些就不多说了。

Virtual Serial Ports Driver XP 5.1 虚拟串口软件，用此软件可以生成一对相互联接的虚拟串口，这样，初期的研究工作就在电脑上完成了，省得用硬件电路板了。

（2）资料

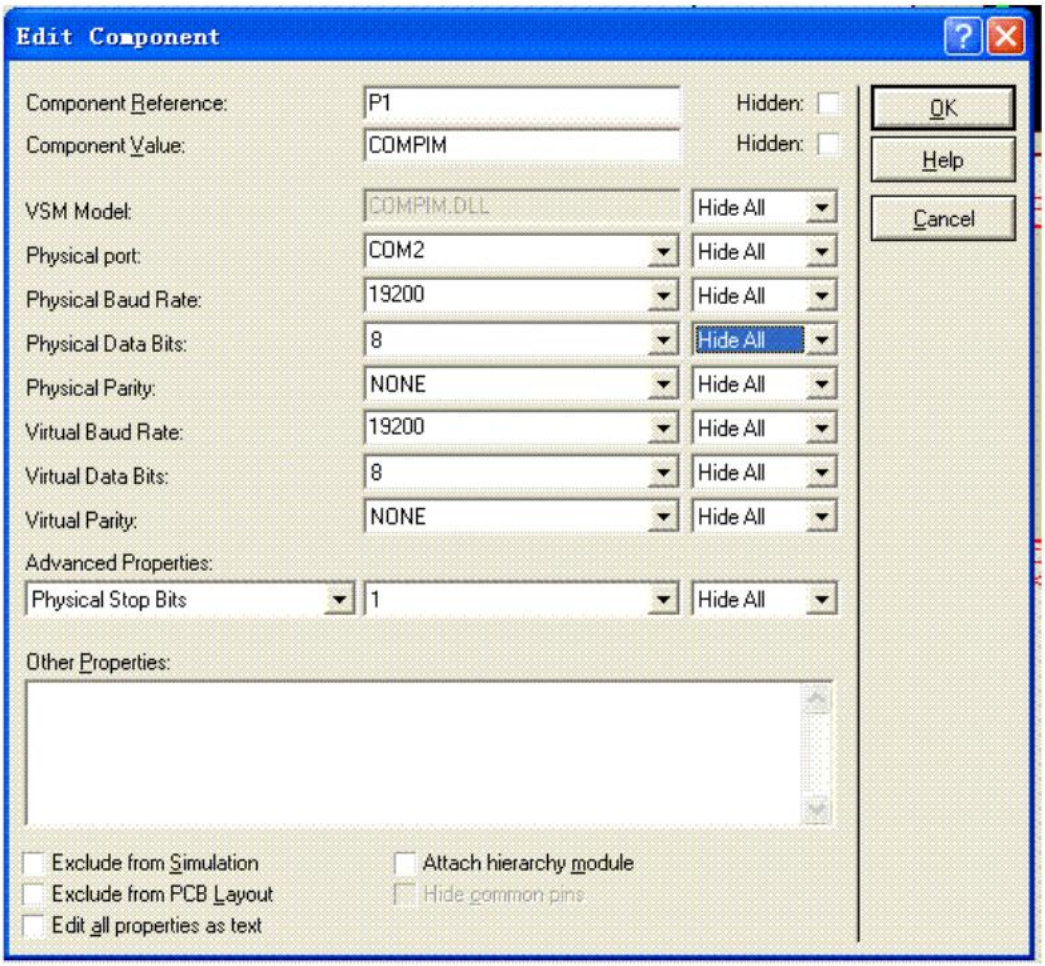
KingView 提供了一份简单的说明材料，就在下载后的解压缩文件包中。具体的位置是：Value Pack\技术资料\常用协议\单片机 ASCII 码通讯协议

（3）电路搭建



注意单片机的 TXD 与虚拟串口的 TXD，单片机的 RXD 与虚拟串口的 RXD 是连在一起的，不要交叉哦，我在这上面可吃了不少的苦头。。。。

这个虚拟串口元件的设置如下图所示：

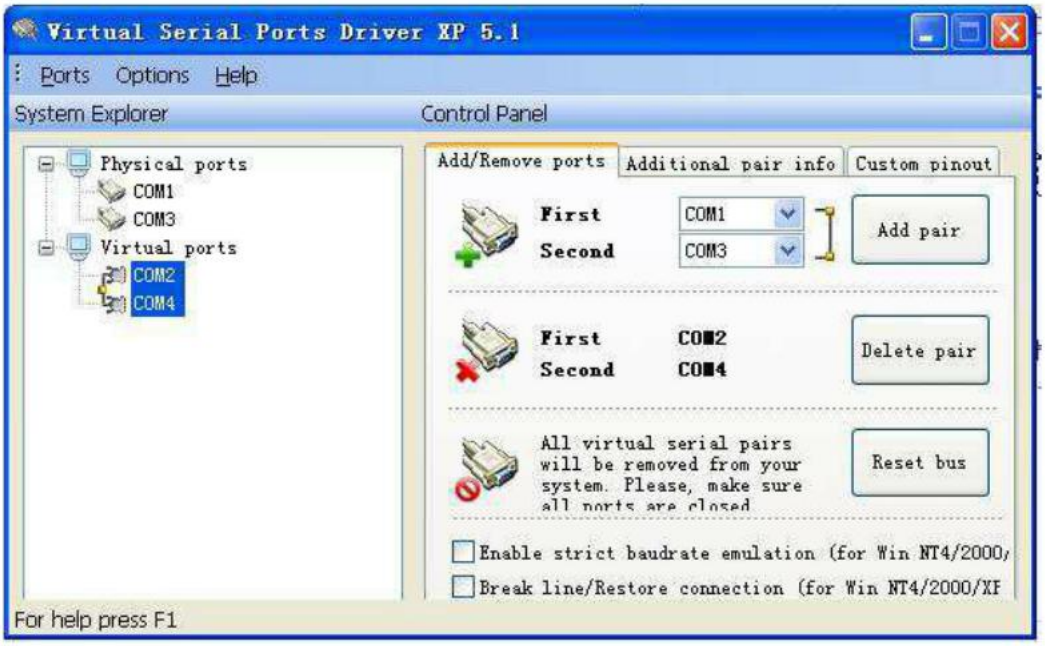


说明：这里选 COM2，是因为我事先用 Vspd 生成了一对虚拟串口，com2 和 com4，至于其他参数则应该选得和 kingview 中的一致，这个到下面再说。

什么，这个元件不知哪里找？这里啦

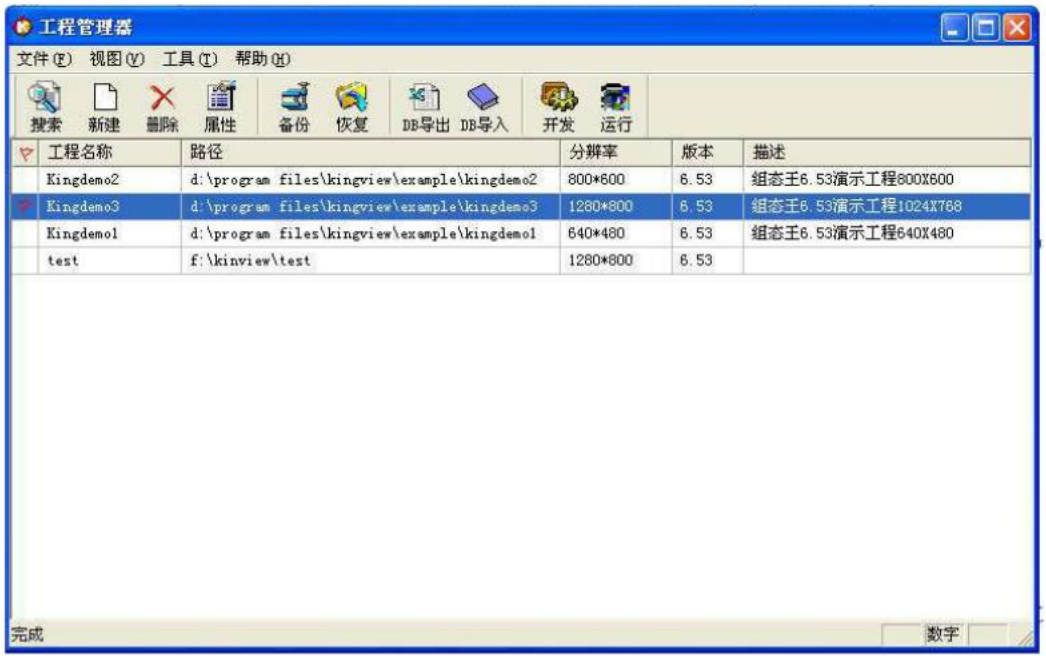
（4）VSPD 的使用

现如今的电脑很少有两个串口的了，人呢也是越来越懒了，虽然手边的电路板是现成的，写片子是容易的，但是仍然还是嫌麻烦的，所以就发动狗狗搜一搜，找到了这个 VSPD，当然它是很容易用的

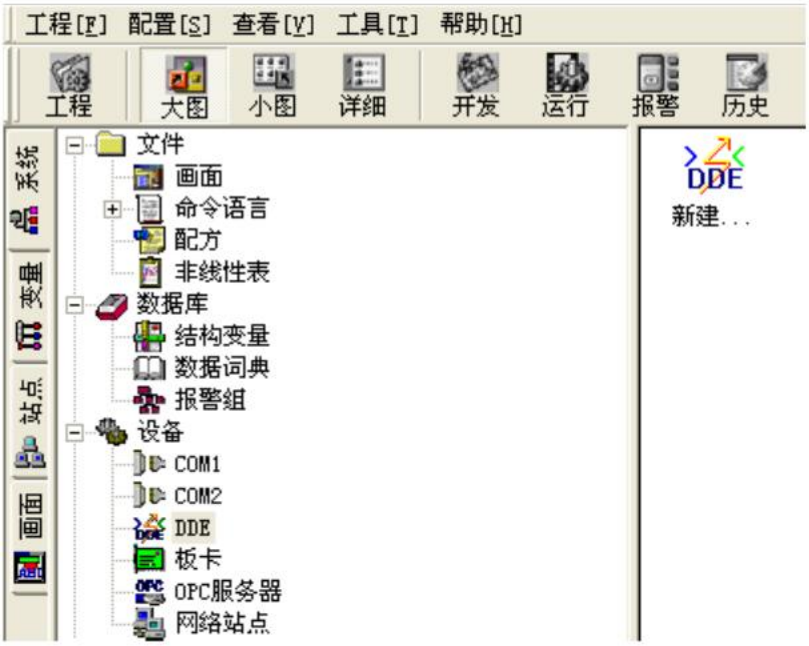


在 first 后面选一个串口名，然后在 Second 后面再选一个串口名，然后点一下 Add Pair 就行啦。怎么选都可以，就算是选 com1 也是可以的，虽然 com1 是真实存在的物理串口，但是这个 VSPD 照样把它给虚拟了。这里我选的是 com2 和 com4，大家可以看到在左侧的窗口中出现了这样一对互联的串口了，也就是说，我从串口 2 发数据，然后串口 4 就能收到。同样，我从串口 4 发数据，串口 2 就能收到。

（5）组态王置



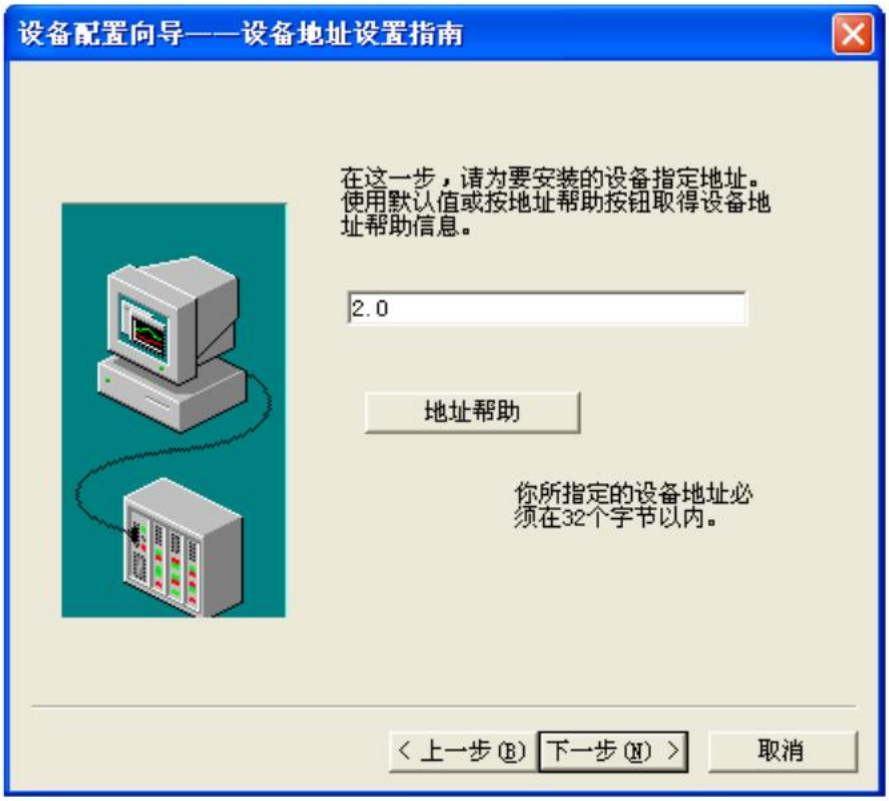
根据自己屏幕选择演示项目中的一个



找到设备->DDE，双击“新建...”

选择：智能模块（上面的图中看不到）->单片机->通用单片机 ASCII->串口

起个名字，然后选择串口号，我们选择 com4



这一步选择地址，需要为自己的单片机设备确定一个地址，这有点麻烦。需要看一看地址帮助，这里简单说明一下。如果在同一个串口上连接多个单片机设备，那么就需要确定究竟与哪一个设备通信，这就需要有个地址，这是上面我取的地址 2.0 中的 2 的由来，而小数点后面可取 0/1，按 kingview 的介绍是打包还是不打包。我还没有理解打包是什么，所以先取 0.

现在“设备”下面多出来了 com4，并且在右侧多出了一个“我的单片机”的图标，这是我为自己的单片机设备起的名字。右击该图标，在弹出的快捷菜单中选择“测试我的单片机”，打开对话框。

在这里选择通信参数，为简单起见，我们将校验选为“无”，其他按图上选择，然后单击“设备测试”进入到设备测试页面。

增加一个寄存器，寄存器 X 后面加个 0，数据类型选择“BYTE，SHORT，FLOAT”三者之一。我们选择 BYTE，选择添加。

OK，至此 kingview 也设置好了。下面就是编程了。

1．通讯口设置：

通讯方式：RS-232,RS-485,RS-422 均可。

波特率：　由单片机决定（2400，4800，9600and19200bps）。

字节数据格式：由单片机决定。

起始位	数据位	校验位	停止位

注意：在组态王中设置的通讯参数如波特率，数据位，停止位，奇偶校验必须与单片机编程中的通讯参数一致

2．在组态王中定义设备地址的格式

格式：＃＃．＃

前面的两个字符是设备地址，范围为 0－255，此地址为单片机的地址，由单片机中的程序决定 ；

后面的一个字符是用户设定是否打包，“0”为不打包、“1”为打包,用户一旦在定义设备时确定了打包，组态王将处理读下位机变量时数据打包的工作。

3．在组态王中定义的寄存器格式

寄存器名称	*dd* 上限	*dd* 下限	数据类型
X*dd*	65535	0	FLOAT/BYTE/UINT

斜体字 *dd* 代表数据地址，此地址与单片机的数据地址相对应。

注意：在组态王中定义变量时，一个 X 寄存器根据所选数据类型（BYTE,UINT,FLOAT）的不同分别占用一个、两个，四个字节，定义不同的数据类型要注意寄存器后面的地址，同一数据区内不可交叉定义不同数据类型的变量。为提高通讯速度建议用户使用连续的数据区。

3．组态王与单片机通讯的命令格式：

读写格式（除字头、字尾外所有字节均为 ASCII 码）

字头	设备地址	标志	数据地址	数据字节数	数据…	异或	CR

说明；

字头：1 字节 1 个 ASCII 码，40H

设备地址： 1 字节 2 个 ASCII 码，0—255（即 0---0x0ffH）

标志：1 字节 2 个 ASCII 码，bit0~bit7，

bit0= 0:读，bit0= 1:写。

bit1= 0：不打包。

bit3bit2 = 00,数据类型为字节。

bit3bit2 = 01,数据类型为字。

bit3bit2 = 1x,数据类型为浮点数。

数据地址： 2 字节 4 个 ASCII 码，0x0000~0xffff

数据字节数：1 字节 2 个 ASCII 码，1—100，实际读写的数据的字节数。

数据…：为实际的数据转换为 ASCII 码，个数为字节数乘 2。

异或：异或从设备地址到异或字节前，异或值转换成 2 个 ASCII 码

CR：0x0d。

有了这些资料，程序就不难编写了。

先测试一下。

到 proteus 中，全速运行，这就打开了串口窗口。

在 kingview 中单击“读取”（见上一篇的最后一个图），可以看到如下字串：

@02E000000176

这个数据字串与地址，数据类型等有关，解读如下：

变量名	类型	字头	设备地址	标志	数据地址	数据字节	异或	CR
X0	BYTE	@	02	E0	0000	01	76	CR
X1	BYTE	@	02	E0	0001	01	77	CR
X0	SHORT	@	02	A4	0000	02	75	CR
X1	SHORT	@	02	A4	0001	02	74	CR
X0	FLOAT	@	02	EC	0000	04	00	CR
X1	FLOAT	@	02	EC	0001	04	01	CR

如果切换成 HEX 显示，则可以看到字头和字

如：

@02A400010274

HEX 显示为：

40 30 32 41 34 30 30 30 31 30 32 37 34 0D

其中取异或的，不包括字头 40H，即从 30H 开始的 10 个字符，异或算出来后，转换成 ASCII 码成为其后的 2 个字符，即 0D 前的两个字符。以上面的数字为例，异或算出来为 74H，转换成 ASCII 码为 37H 和 34H。

不多说啦，上一个写好的程序：

```
#include "reg52.h"
/*11.0592M
19200 bps
*/
typedef unsigned char uchar;
typedef unsigned int uint;
/*定时器 2 的波特率：fosc/32*(65536-(rcap2h rcap2l))
按此，可得波特率是 19。2 时，要求 65536-(rcap2h rcap2l)=18
即（rcap2h rcap2l）=65518
*/
void serial_init ()  {
   SCON  = 0x50;        /* mode 1: 8-bit UART, enable receiver   */
   C_T2=0;     /*Timer2 runing in Timer mode*/
    RCLK="1";
    TCLK="1";
   RCAP2H=0xff;
   RCAP2L=0xee;
 TR2=1;   /*enable Timer2 run  */
   ES    = 1; REN="1"; EA="1"; SM2=1;      /*SM2=1 时收到的第 9 位为 1 才置位 RI 标志*/
}
/*通过串行口发送数据  */
void UartSend(uchar Dat)
{   SBUF=Dat;
 for(;;)
 { if(TI)
   break;
 }
 TI=0;
}
uchar RecBuf[12];
bit   RecOk="0"; //一次接收工作结束
void Recive() interrupt 4
{
 static bit StartRec="0";   //如果收到的首个字符是 40H，则该值取 0
 static uchar Count="0"; //计数器
 uchar RecDat;
 RecDat=SBUF;   //取得 SBUF 中的数据
 if(!StartRec)   //新的一次接收开始
 {
  if(RecDat==0x40)  //首字符正确
  {
```

```
   StartRec=1;  //开始新的一次接收工作
  }
 }
 else
 {
  RecBuf[Count]=RecDat;
  Count++;
  if(RecDat==0x0d)
  {
   StartRec=0; //准备下一次接收
   Count=0;  //计数器清零
   RecOk=1;  //一次接收正确
  }
 }
 RI=0;
}
void UartSends(uchar Buff[],uchar Len)
{ uchar i;
 for(i=0;i<Len;i++)
 { UartSend(Buff[i]);
 }
}
void mDelay(uint DelayTim)
{
 uchar i;
 for(;DelayTim>0;DelayTim--)
 {
  for(i=0;i<123;i++);
 }
}
uchar SendBuf[10]={0x40,0x30,0x33,0x30,0x31,0x36,0x35,0x30,0x31,0x0d};
void main()
{ uchar i;
 uchar RecCount="0";
 uchar SendCount="0";
 uchar xorDat;
 uchar cTmp1,cTmp2;
 uchar cTmp;
 uchar SendDat="1";//这个是程序中准备传递给 kingview 的，可以自行更改。
 serial_init (); //定时器，串口初始化
 for(;;){
    if(RecOk)      //一次完整的接收
  {
   RecOk=0;      //本次接收后的应答处理完毕
   xorDat=RecBuf[0];
```

```
    for(i=1;i<10;i++)
    {
     xorDat^=RecBuf[i]; //异或
    }
    cTmp1=xorDat&0xf0;  //取高 4 位
    cTmp1>>=4;     //右移 4 次移到低 4 位
    cTmp1+=0x30;
    cTmp2=xorDat&0x0f;  //取低 4 位
    cTmp2+=0x30;
    if((cTmp1==RecBuf[10])&&(cTmp2==RecBuf[11]))
    {   SendBuf[1]=RecBuf[0];SendBuf[2]=RecBuf[1]; //地址与接收到的地址相同
     //SendBuf[3]=  SendBuf[4]= //发送的字节数
     cTmp=SendDat;
     cTmp&=0xf0;  //取高 4 位
     cTmp>>=4;  //右移 4 位
     SendBuf[5]=0x30+cTmp;
     cTmp=SendDat;
     cTmp&=0x0f;
     SendBuf[6]=0x30+cTmp;//发送的值
     //////以下计算异或码
     xorDat=SendBuf[1];
     for(i=2;i<7;i++)
     {
      xorDat^=SendBuf[i];
     }
     cTmp1=xorDat&0xf0;  //取高 4 位
     cTmp1>>=4;
     cTmp1+=0x30;
     cTmp2=xorDat&0x0f;  //取低 4 位
     cTmp2+=0x30;
     SendBuf[7]=cTmp1;SendBuf[8]=cTmp2;
     //做异或运算
       UartSends(SendBuf,10);
    }
   }
  }
}
```