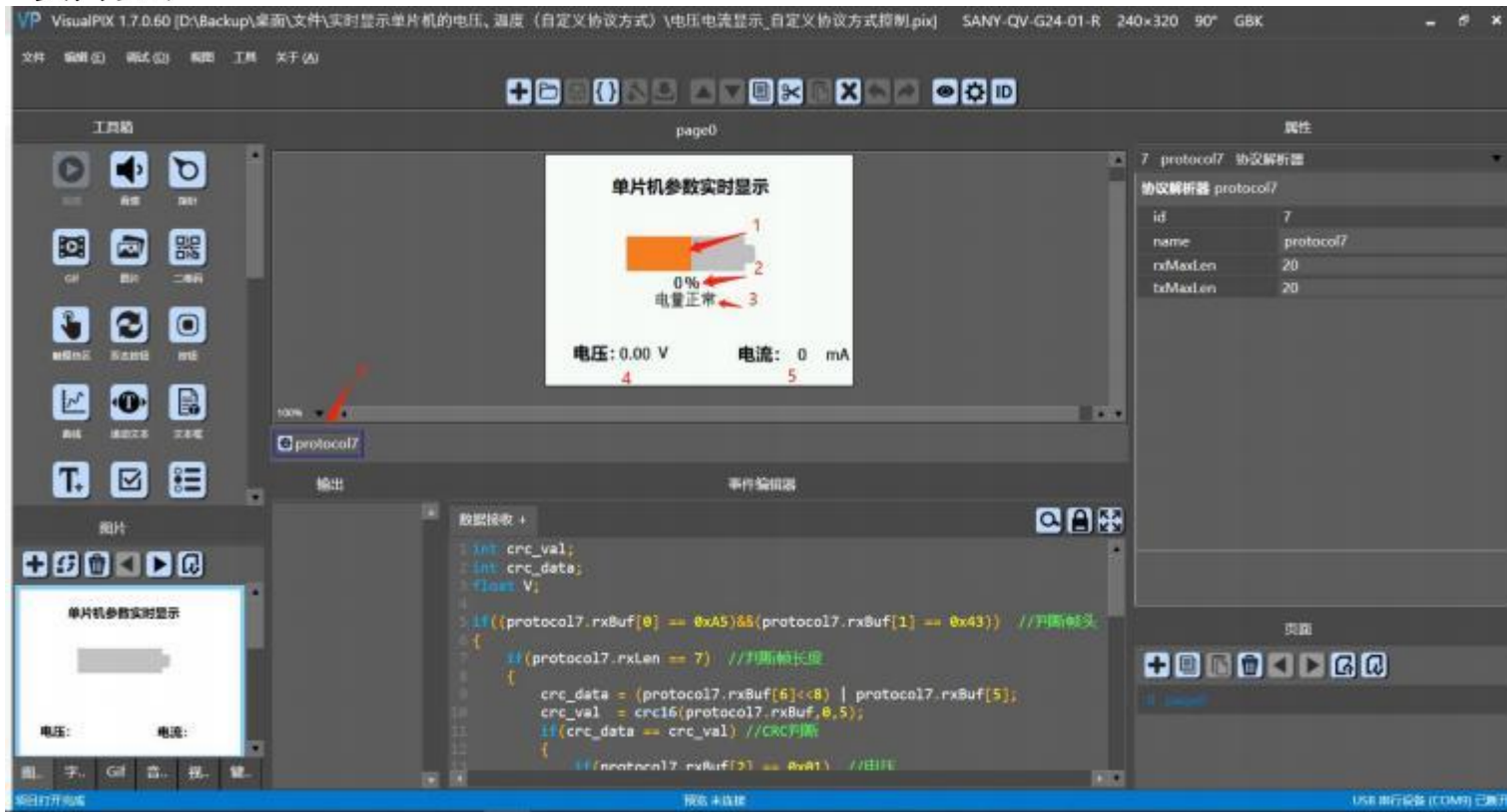


## 实验 2 实时显示单片机的参数（整数、小数、中文 自定义协议方式）

### 1.实验目的

单片机按照自定义协议格式发送控制命令，串口屏实时显示单片机的参数，包括整数、小数、和中文字符。实现效果和 实验一相同。

### 2.页面设计



- (1) 度条控件，以进度条的方式显示出SOC 值
- (2) 整数字件，显示具体 SOC 值
- (3) 文本控件，显示电量状态
- (4) 浮点数字件，显示电压值
- (5) 整数字件，显示电流值
- (6) 协议解析器控件，编写脚本代码，解析自定义的协议

选中控件，可以在右侧属性栏查看控件的各个属性。详细属性说明可参考 第四章 控件的介绍和用法。

资源区：

一共使用4 张图片，home 作为页背景图片，另外三张图片为进度条不同状态的颜色。

字体除了工程自带的字库 Tahoma\_4x\_ASCII （包括英文和数字），另外创建了一个中文的字库，双击这个字库，可以看到 详细的信息。字库详细制作方法可以参考第三章的添加字库小节。

## 3. 串口屏协议处理

该例程使用了协议解析器控件 。串口屏每次收到一帧数据，都会触发一次协议解析器的脚本，和单片机的空闲中断函数类似。

协议定义（仅仅是示例，仅供参考）

帧头(2 字节) 命令码(1 字节) 数据(2 字节) 校验(2 字节)

A5 43 01 2byte 2byte //设置电压 使用时除 100，如 3.14V，传输时是 314

A5 43 02 2byte 2byte //设置电流

A5 43 03 2byte 2byte //设置 SOC

数据和校验 使用小端模式，即低字节在前，高字节在后

校验方式为 CRC16，多项式 8005

选中控件，在事件编辑器串口可以看到数据的解析过程，代码如下：

```
int crc_val;
int crc_data;
float V;

if((protocol7.rxBuf[0] == 0xA5)&&(protocol7.rxBuf[1] == 0x43)) //①判断帧头
{
    if(protocol7.rxLen == 7) //②判断帧长度
    {
        crc_data = (protocol7.rxBuf[6]<<8) | protocol7.rxBuf[5];
        crc_val = crc16(protocol7.rxBuf,0,5);//③
        if(crc_data == crc_val) //CRC 判断
        {
            if(protocol7.rxBuf[2] == 0x01) //电压
            {
                V = (protocol7.rxBuf[4]<<8) | protocol7.rxBuf[3];
                numVf.valf = V/100;
            }
            else if(protocol7.rxBuf[2] == 0x02) //电流
            {
                numA.val = (protocol7.rxBuf[4]<<8) | protocol7.rxBuf[3];
            }
            else if(protocol7.rxBuf[2] == 0x03) //SOC
            {
                psoc.val = (protocol7.rxBuf[4]<<8) | protocol7.rxBuf[3];
                nsoc.val = psoc.val;

                if(nsoc.val > 80)
                {
                    psoc.fglmg = user.image.blue; ④
                    text.txt = "电量充足";
                }
            }
            else if(nsoc.val > 20)
```



```
{
    psoc.fgImg = user.image.yellow;
    text.txt = "电量正常";
}
else
{
    psoc.fgImg = user.image.red;
    text.txt = "电量过低";
}
}
}
```

- ① 接收到串口数据缓存在协议解析器控件的 `rxBuf` 里，是一个数组，数组长度是协议解析器的一个属性，可配置。决定接收一帧数据的最大长度，最大可设置 512 字节。
- ② 属性 `rxLen` 是本次接收到数据长度。
- ③ `crc16` 是一个内部函数，用于计算 CRC 校验值。具体使用方法 请参考 第五章 函数的介绍和用法。具体项目中，可选择 使用或者不使用校验。
- ④ `psoc.fgImg = user.image.blue`;这里是修改进度条的图片，也可以写成这样 `psoc.fgImg = 2`;效果是一样的。

## 4. 下载验证

编译成功后，点击下载按钮，选择正确的端口号和波特率，下载到串口屏。可以看到显示如下显示：





注意：发送时勾选 hex 发送

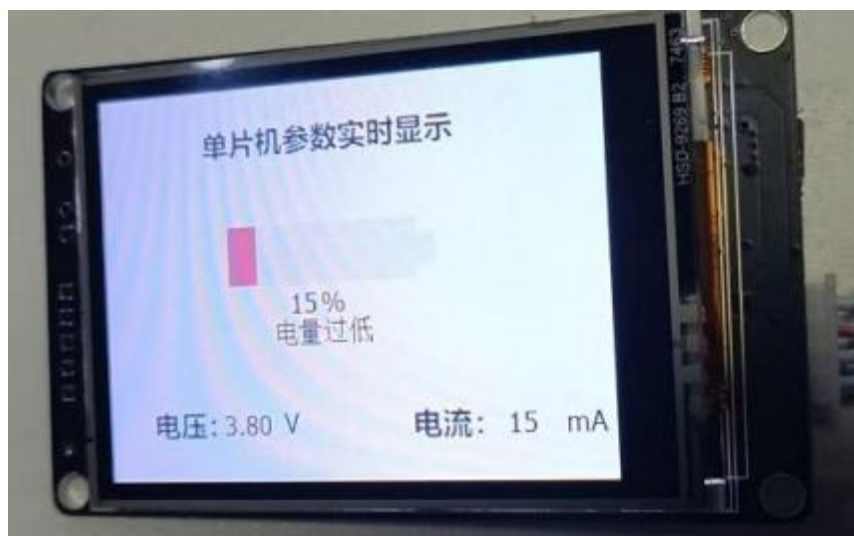
测试的命令

A5 43 01 7C 01 91 86 //设置电压值 3.80V

A5 43 02 0F 00 AE 2C //设置电流值 15mA

A5 43 03 0F 00 B9 AC //设置 SOC 值 15%

串口屏显示如下，表示串口屏收到数据并正确处理。



实际应用时，是 MCU 和串口屏通信。本实验使用的单片机型号是 STM32F103TB,改例程仅作参考，开发环境 keil V5.24 。其 他 MCU 的串口发送方式也类似。



部分代码

```
int main(void)
{
    u16 crc_val;

    Sys_Set Rcc();                //设置系统使用内部时钟
    delay_init(64);               //延时函数初始化
    usart1_init();                 //初始化串口 波特率 115200

    while(1)
    {
        //设置电压值
        str[0] = 0xA5;
        str[1] = 0x43;
        str[2] = 0x01;
        str[3] = (V>>0) & 0xFF;
        str[4] = (V>>8) & 0xFF;
        crc_val = Get_CRC16(5,str);
        str[5] = (crc_val>>0) & 0xFF;
        str[6] = (crc_val>>8) & 0xFF;

        USART_OUT( USART1, (uint8_t*)str,7);
        delay_ms(1); //命令之间需要加上小延时分开

        //设置电流值
        str[0] = 0xA5;
        str[1] = 0x43;
        str[2] = 0x02;
        str[3] = (I>>0) & 0xFF;
        str[4] = (I>>8) & 0xFF;
        crc_val = Get_CRC16(5,str);
        str[5] = (crc_val>>0) & 0xFF;
        str[6] = (crc_val>>8) & 0xFF;

        USART_OUT( USART1, (uint8_t*)str,7);
        delay_ms(1); //命令之间需要加上小延时分开

        //设置 SOC
        str[0] = 0xA5;
        str[1] = 0x43;
        str[2] = 0x03;
        str[3] = (SOC>>0) & 0xFF;
        str[4] = (SOC>>8) & 0xFF;
        crc_val = Get_CRC16(5,str);
        str[5] = (crc_val>>0) & 0xFF;
```



```
str[6] = (crc_val>>8) & 0xFF;

USART_OUT( USART1, (uint8_t*)str,7);
delay_ms(1); //命令之间需要加上小延时分开

//模拟数据变化
V+=8;
if(V > 500)V = 300; //电压值范围 3~5V
if(++I > 100)I = 0;
if(++SOC > 100)SOC = 0;

delay_ms(200);

}
```

单片机部分代码很简单，按照协议格式组帧发送即可。注意每条命令之间需要加一点点延时用于串口屏区分。

总结：

例程二和例程一，最后的显示结果都是一样的。系统指令的方式适合界面不复杂的情况，无需设计通信协议，按照串口屏内存的系统指令格式来发送即可；自定义协议方式体现为更加灵活，C语言编写脚本，通信效率更高，一帧数据可给多个控件赋值，适用于大多数项目。推荐使用自定义协议方式。两种方式混合使用也可以。