



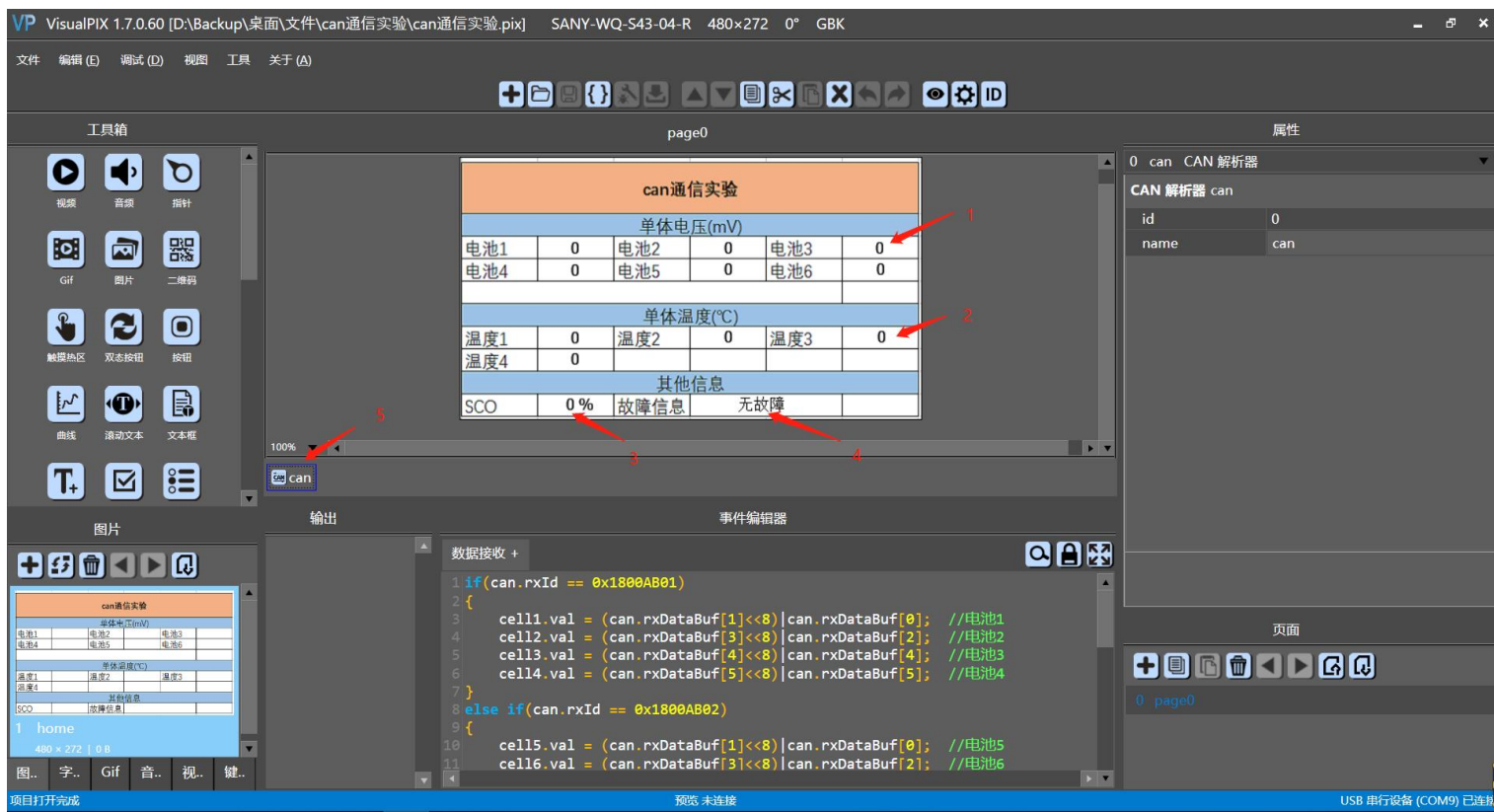
实验 5 can 通信实验

1. 实验目的

按照指定协议，解析 can 报文，显示电压温度等信息。

说明：4.3 寸及以上带 can 版本的串口屏，支持 can 通信。

2. 页面设计



- (1) 整型控件，显示电压值
- (2) 整型控件，显示温度值
- (3) 整型控件，显示 SOC
- (4) 文本控件，显示故障信息
- (5) can 协议解析器控件，可以编写脚本代码，解析 can 报文。

图片资源，使用了一个 480*272 分辨率的背景图片。页面的背景图片额可以使用专业的绘图软件更加丰富、绚烂的背景图片。图片素材决定了显示的整体外观，让界面的设计更加容易。

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 can 解析器。串口屏每次收到一帧 can 数据，都会触发一次 can 解析器的脚本，和单片机的 can 中断函数类似。

协议定义（仅供参考）

扩展数据帧，波特率 250K		
ExdId	data[0~7]	
0x1800AB01	[0~1] 电池 1 电压值 [2~3] 电池 2 电压值 [4~5] 电池 3 电压值 [6~7] 电池 4 电压值	小端模式，单位 mV
0x1800AB02	[0~1] 电池 5 电压值 [2~3] 电池 6 电压值	小端模式，单位 mV
0x1800AB10	[0] 温度 1 [1] 温度 2	小端模式，单位℃
0x1800AB11	[1] 温度 3 [2] 温度 4	小端模式，单位℃
0x1800AB12	[0] SOC	范围 0~100
0x1800AB13	[0] 故障信息 00 无故障 01 电池故障 02 温度故障 03 系统故障	

选中 can 解析器，可看到事件编辑窗口，事件代码如下：

```
if(can.rxlId == 0x1800AB01)
{
    cell1.val = (can.rxDataBuf[1]<<8)|can.rxDataBuf[0]; //电池 1
    cell2.val = (can.rxDataBuf[3]<<8)|can.rxDataBuf[2]; //电池 2
    cell3.val = (can.rxDataBuf[4]<<8)|can.rxDataBuf[4]; //电池 3
    cell4.val = (can.rxDataBuf[5]<<8)|can.rxDataBuf[5]; //电池 4
}
else if(can.rxlId == 0x1800AB02)
{
    cell5.val = (can.rxDataBuf[1]<<8)|can.rxDataBuf[0]; //电池 5
    cell6.val = (can.rxDataBuf[3]<<8)|can.rxDataBuf[2]; //电池 6
}
else if(can.rxlId == 0x1800AB11)
{
    t1.val = can.rxDataBuf[0]; //温度 1
    t2.val = can.rxDataBuf[1]; //温度 2
    t3.val = can.rxDataBuf[2]; //温度 3
```



```
t4.val = can.rxDataBuf[3]; //温度 4
}
else if(can.rxId == 0x1800AB12)
{
    soc.val = can.rxDataBuf[0]; //SOC
}
else if(can.rxId == 0x1800AB13) //故障信息
{
    if(can.rxDataBuf[0] == 0x00)
        fault.txt = "无故障";
    else if(can.rxDataBuf[0] == 0x01)
        fault.txt = "电池故障";
    else if(can.rxDataBuf[0] == 0x02)
        fault.txt = "温度故障";
    else if(can.rxDataBuf[0] == 0x03)
        fault.txt = "系统故障";
}
```

事件解析器的代码很简单，can.rxId 存放报文 id, can.rxDataBuf 存放数据，数组长度 8 字节。按照协议解析数据即可。

4. 硬件过滤器的配置

显示屏接入 can 总线时，可以设置硬件过滤器。硬件上过滤掉不需要接收的报文，减少显示屏处理过多无用报文的压力，提高显示屏刷屏流畅度。设计 can 程序时，配置过滤器很有必要。

显示屏 can 收发调试过程中，可以不配置过滤器，方便调试。工程调试完成后，再将掩码和滤码设置成合适的值。显示屏过滤器最多有 14 个，编号 0~13，每个过滤器有两种过滤模式：屏蔽位模式、标识符列表模式

屏蔽位模式：

需要设置过滤标识符 filter 和过滤掩码 mask。硬件收到一帧 can 报文 id 时，按照 mask 为 1 的 bit 与过滤标识符对比，全部 mask 位相同时，则接收该报文，否则拒收。换个说法：如果 $(id \& mask) == filter$ ，接收此报文；否则不符合过滤要求，不接收此报文。适用于要接收 can 报文比较多且报文 id 有一定规律。

标识符列表模式：

硬件收到一帧 can 报文 id 时，与过滤器列表一一对应，如果有匹配的报文 id 则接收该报文，否则拒收。每个过滤器可以设置两个 ID，所以最多可设置 28 个。适用于要接收的 can 报文比较少少的情况，可以实现 can 报文的精准过滤。

串口屏的过滤器设置可以通过项目设置界面配置(只能配置过滤器 0 的屏蔽位模式)，也能在脚本里调用 can 控件的方法 setFilter 来配置(通过参数来配置过滤器的参数，可以灵活的设置的工作模式)。

1、项目设置界面：

打开 "can 通信实验.pix" 工程，点击项目设置图标，可以查看到 can 的波特率、过滤报文的掩码和滤码。



这种配置方式，只能设置过滤器 0 的屏蔽位模式。适用于过滤器简单的设置，优点是配置简单

计算掩码 mask 和滤码 filter 的时候，考虑所有要接收的报文 id，找出相同的 bit 和有差异的 bit。该例程的所有要接收的报文有 6 个 id，如下：

0x1800AB01
0x1800AB02
0x1800AB10
0x1800AB11
0x1800AB12
0x1800AB13

有差异的 bit 有[1:0]、[4]，其余 bit 都是相同的。则 mask 的[1:0]、[4]为 0，其他为 1，即 mask=0xFFFFFEB。相同的 bit 即为 filter，即 filter=0x1800AB00。最后可将上面 5 个验证 是否 (id & mask) == filter。

2、调用 can 控件的方法 setFilter

当过滤的规则比较复杂时，建议使用 setFilter 方法来配置过滤器。

打开 “can 通信实验_setFilter.pix”工程，点击 page0 空白处，可以看到加载页面事件，脚本代码如下

```
加载页面 + 离开页面
1 //用startup标记, 上电只执行一次
2 if(startup.val == 1)
3 {
4     startup.val = 0;
5
6     //配置过滤器
7
8
9     //标识符列表模式
10    can.setFilter(0,1,(0x1800AB01<<3)|0x04,(0x1800AB02<<3)|0x04);
11    can.setFilter(1,1,(0x1800AB10<<3)|0x04,(0x1800AB11<<3)|0x04);
12    can.setFilter(2,1,(0x1800AB12<<3)|0x04,(0x1800AB13<<3)|0x04);
13
14
15
16
17    //屏蔽位模式
18    //can.setFilter(0,0,(0xFFFFFEB<<3)|0x04,(0x1800AB00<<3)|0x04);
19
20 }
```

这里需要注意的是，该脚本是首页的加载页面事件，用了一个变量 startup 来标记该脚本只执行一次，因为配置过滤器只需上电配置一次就可以了。



```
void setFilter(u8 filterid,u8 mode,u32 param1, u32 param2);  
/*  
函数功能：配置 can 接收过滤器  
参数 1: filterid,过滤器编号，0~13  
参数 2: mode,过滤器模式，0 屏蔽位模式，1 标识符列表模式  
参数 3: param1,过滤器参数 1，参数位定义见下表  
参数 4: param2,过滤器参数 2，参数位定义见下表  
*/
```

过滤器 param1 和 param2 位定义：

位 31:21	STID[10:0]/EXID[28:18]:标准标识符或扩展标识符 依据 IDE 位的内容，这些位 或是标准标识符(11bit)，或是扩展标识符的高 11bit
位 20:3	EXID[17:0]扩展标识符低 18bit
位 2	IDE:标识符类型 0: 标准帧 1: 扩展帧
位 1	RTR:远程发送请求 0: 数据帧 1: 远程帧
位 0	保留位

标识符列表模式

```
can.setFilter(0,1,(0x1800AB01<<3)|0x04,(0x1800AB02<<3)|0x04); //过滤器 0  
can.setFilter(1,1,(0x1800AB10<<3)|0x04,(0x1800AB11<<3)|0x04); //过滤器 1  
can.setFilter(2,1,(0x1800AB12<<3)|0x04,(0x1800AB13<<3)|0x04); //过滤器 2
```

屏蔽位模式

```
can.setFilter(0,0,(0xFFFFFEB<<3)|0x04,(0x1800AB00<<3)|0x04); //过滤器 0
```

方法 setFilter 的参数 param1 和 param2 都与 0x04 按位或运算，表示只接收扩展帧。

注：
假如要接收标准帧，如 0x6A1(标准帧)，0x6A2(标准帧)，可以这样设置：

```
CAN_Set_Filter2(0,1,0x6A1<<21,0x6A2<<21); 过滤器 0，标识符列表模式
```

或者：

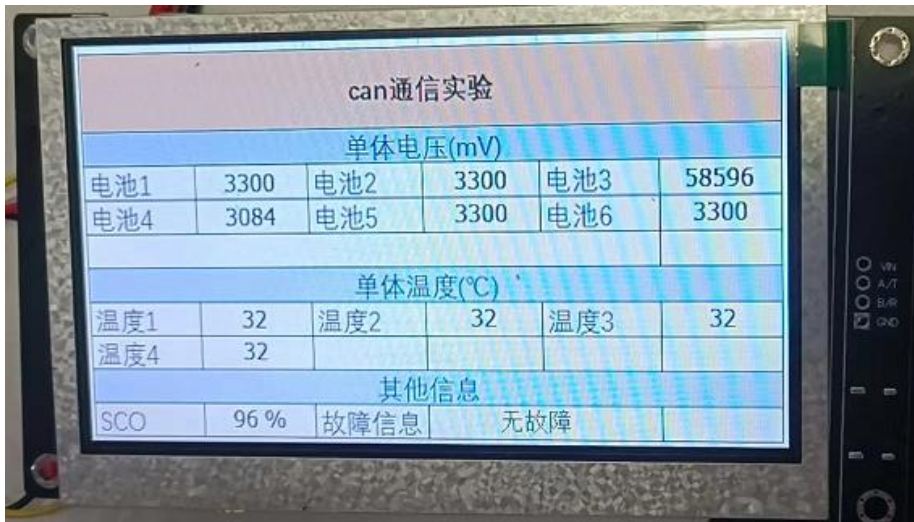
```
CAN_Set_Filter2(0,0,0xFFC<<21,0x6A0<<21); 过滤器 0，屏蔽位模式
```

至于这里为什么是左移 21 位，可以参考上表 param1 和 param2 位定义。

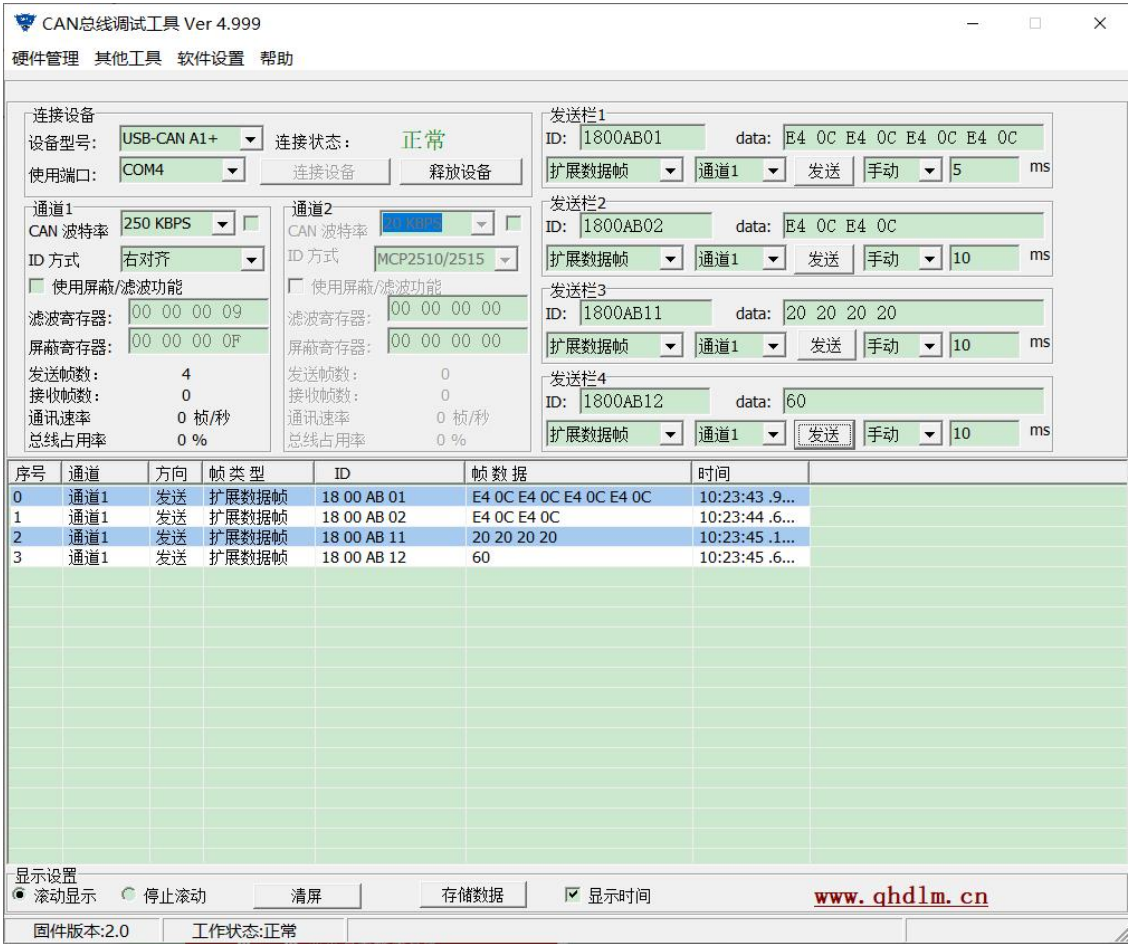
5. 下载验证

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。将串口屏接到 can 总线上，串口屏收到正确报文时，会解析并显示。

调试时，用任意 can 收发器的助手，给显示屏发报文测试即可。界面如下：



can 调试工具发送数据



测试命令如下:

报文 ID	数据	说明
1800AB01	E4 0C E4 0C E4 0C E4 0C	设置电池 1~4
1800AB02	E4 0C E4 0C	设置电池 5~6
1800AB10	20 20	设置温度 1~2
1800AB11	20 20	设置温度 3~4
1800AB12	60	设置 SOC
1800AB13	00	故障信息