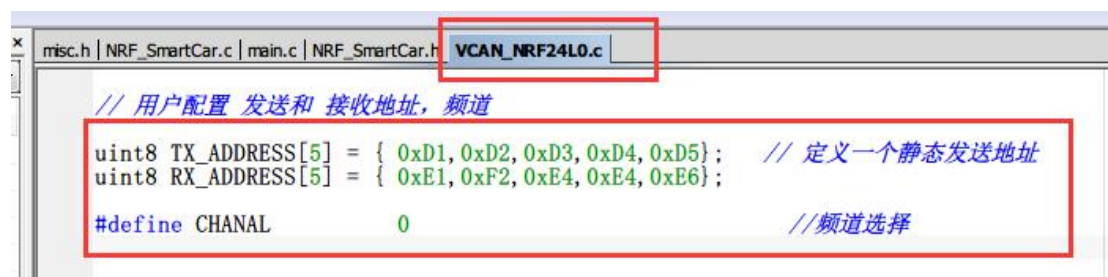


智能车调试平台使用手册

注意：

- 1.使用前提是 **NRF** 已经调通，芯片已经能和电脑进行通讯。
- 2.收发的数据精确到小数点后 **3** 位，整数部分最大为 **65535**。

若之前不能通讯，下位机需要改动的地方为收发地址和收发频率，如下所示。



```
misc.h | NRF_SmartCar.c | main.c | NRF_SmartCar.h | VCAN_NRF24L01.c
// 用户配置 发送和 接收地址，频道
uint8 TX_ADDRESS[5] = { 0xD1, 0xD2, 0xD3, 0xD4, 0xD5}; // 定义一个静态发送地址
uint8 RX_ADDRESS[5] = { 0xE1, 0xF2, 0xE4, 0xE4, 0xE6};
#define CHANAL          0 //频道选择
```

一、功能介绍

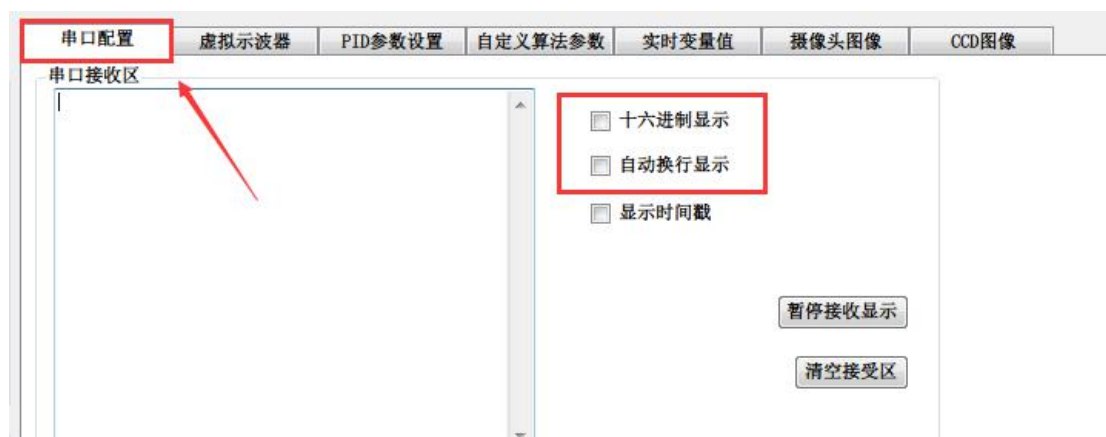
1. **串口配置**：缩减版的串口调试助手，ASCII 模式下只能显示英文和数字。
2. **虚拟示波器**：实时显示一些参数的变化情况
3. **PID 参数设置**：修改四轮车和平衡车的 PID 参数
4. **自定义算法参数**：修改自己新加的算法的参数，作为 PID 参数设置的扩展版本。
5. **实时变量值**：功能和 IAR 下加断点调试相似，刷新频率取决于下位机对应部分的发送频率。
6. **摄像头图像**：只能显示山外鹰眼摄像头压缩后的数据，图像尺寸是 600。
7. **CCD 图像**：只能显示山外 CCD 的图像，图像尺寸 128，支持同时 3 路 CCD 一起显示。
8. **全局参数设置**（左下角位置）：
 1. **读取全局参数**：获取下位机当前对应变量的值。
 2. **紧急停车**：停止当前车模运动
 3. **保存当前运行参数**：将虚拟示波器中曲线，PID 参数设置和自定义算法参数中的数据保存到一个 ini 配置文件中。
 4. **导入历史数据**：还原之前保存的参数。
 5. **数据导出到 matlab**：这个功能未写。

二、串口设置

按照一般串口调试助手的方式配置好，主要是串口号和波特率，需要和 NRF 转 USB 模块对应上。



1. 串口配置，如下图：



十六进制显示和自动换行显示不勾选，其他的不用修改。配置完毕。

二、虚拟示波器

若需要在上位机显示，需要在下位代码中写上如下其中一条即可。

比如，需要在上位机**第一路**显示，需在下位机对应位置写：

```
SendPack_Short(RealTime,RealTime_1, (int)1, 1, 1);
```

其中的“(int)1”是你需要在上位机显示的变量；“(int)1”后面的两个 1，保持不变。

```

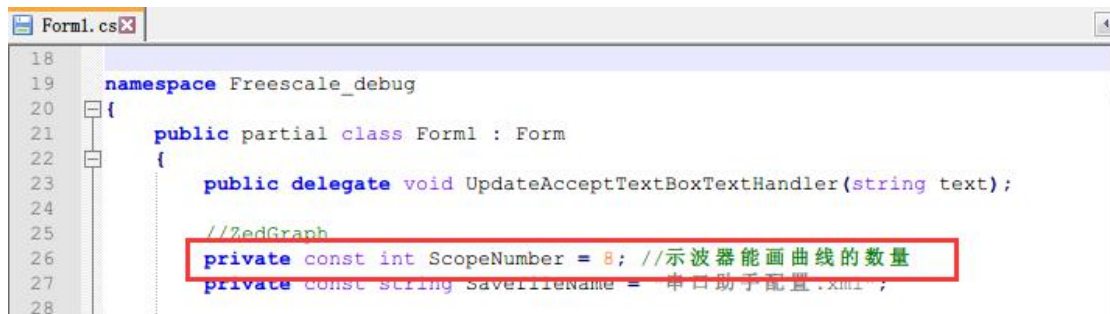
//*****
//实时参数
//SendPack_Short(Electricity,Electricity_1, /*你的变量名*/ 0, 1, 1);
//*****
//实时参数
SendPack_Short(RealTime,RealTime_1, (int)1, 1, 1); //第一路
SendPack_Short(RealTime,RealTime_2, (int)2, 1, 1); //第二路
SendPack_Short(RealTime,RealTime_3, (int)22, 1, 1); //第三路
SendPack_Short(RealTime,RealTime_4, (int)33, 1, 1);
SendPack_Short(RealTime,RealTime_5, (int)45, 1, 1);
SendPack_Short(RealTime,RealTime_6, (int)33, 1, 1);
//*****
//=====

```

假设需要在**第二路**显示（注意加粗部分），就写：

```
SendPack_Short(RealTime,RealTime_2, (int)1, 1, 1);
```

注意：**RealTime_***的最大值是**8**，最多只能**8**路同时显示，若需要再增加或减少显示路数，需要修改上位机 Form1.cs 文件中的 **ScopeNumber** 变量即可。如下所示：



三、PID 参数设置

PID 参数主要有 5 种：

- 舵机 PID
- 四轮车的电机 PID
- 直立车直立 PID
- 直立车速度 PID
- 直立车方向 PID

只介绍舵机 PID 的配置方法，需要修改下位机对应代码，需要注意的地方如下图标注，有四个。其中

SPEED_CONTROL_P/SPEED_CONTROL_I/SPEED_CONTROL_D 是和你代码中和舵机 PID 相关的变量。

对于其他种类的 PID，只需要把取消注释，把 PID 部分变量名对应上即可。

因 PID 中存在 I，即积分环节，修改 PID 参数的同时推荐在里面把积分部分清理，请自行设定。

```
//在这里设置接收到的设置参数（PID和DIY参数）
//=====
//舵机
if(_pidsettings[0].FlagValueChanged == 1)
{
    /*在这里输入你的变量名——下同*/
    SPEED_CONTROL_P = _pidsettings[0].Value_P;
    SPEED_CONTROL_I = _pidsettings[0].Value_I;
    SPEED_CONTROL_D = _pidsettings[0].Value_D;
    _pidsettings[0].FlagValueChanged = 0;
    SendPack_Echo(SendBackPID_ACK, Servo_ACK, "ACK");
}
//=====
```

四、自定义算法参数

自定义算法参数对应目前有 16 路。

若要使用“自定义参数一”，只需要在如下框架下，把 CAR_SPEED_SET 更改为你需要的变量即可，支持 float 类型，精确到小数点后三位。SendPackEcho 的作用是在改完参数后，给电脑发送一个指令，表明已经改完参数。DIY_Para_1_ACK 表示是第一路的确认信号，DIY_Para_2_ACK 是第二路的确认信号，以此类推。

下位机“自定义参数一”的代码，与上位机的这个部分一一对应。

```
////////////////////////////////////
//自定义参数一
if (_diyparameter[0].FlagValueChanged == 1)
{
    CAR_SPEED_SET = _diyparameter[0].DIY_Value;
    _diyparameter[0].FlagValueChanged = 0;
    SendPack_Echo(SendBackDIY_ACK, DIY_Para_1_ACK, "ACK");
}
//=====
//自定义参数二
if (_diyparameter[1].FlagValueChanged == 1)
{
    diyTest = _diyparameter[1].DIY_Value;
    _diyparameter[1].FlagValueChanged = 0;
    SendPack_Echo(SendBackDIY_ACK, DIY_Para_2_ACK, "ACK");
}
}
```

下位机代码

Checkbox	ID	Name	Value	Modify
<input checked="" type="checkbox"/>	1	ghg	10.5	修改
<input checked="" type="checkbox"/>	2	Name2	100	修改
<input checked="" type="checkbox"/>	3	Name3	10.2	修改
<input type="checkbox"/>	4	Name4	1.0	修改
<input type="checkbox"/>	5	Name5	1.0	修改
<input type="checkbox"/>	6	Name6	1.0	修改
<input type="checkbox"/>	7	Name7	1.0	修改

上位机界面

若要使用“自定义参数二”，请注意_diyparameter 数组的下标，参数一的时候数组下标是 0，参数二的时候数组下标是 1。

五、实时变量值

函数中第一个和第二个参数代表的是实时变量。使用方法和“四、虚拟示波器”中的相似，只不过形参变成了 Electricity 和 Electricity_1。

```

//*****
//实时参数
//SendPack_Short(Electricity, Electricity_1, /*你的变量名*/ 0, 1, 1);

```

六、摄像头图像

使用时

1.上位机需要先勾选上，如下所示。



2.把山外摄像头数据采集的代码配置好，再在下位机中把这个的注释取消即可。主要是关注后面两个形参，imgbuff 是经过压缩的摄像头数据，CAMERA_SIZE 是 600。

```
//=====
//摄像头图像
//SendPack_Camera(Camera, Camera_1, imgbuff, CAMERA_SIZE);
```

七、CCD 图像

1.使用时，上位机需要先勾选上，如下所示。



2.把山外关于 CCD 数据采集的代码配置好，再取消这个注释。

代码中，第二个形参表示是第几个 CCD，CCD_BUFF[*]里是对应的 CCD 数据，其他参数不用关注。

```
//*****
//CCD图像
//SendPack_CCD(2, 1, (uint8_t *)&CCD_BUFF[0], TSL1401_SIZE, 1, 1);
//SendPack_CCD(2, 2, (uint8_t *)&CCD_BUFF[1], TSL1401_SIZE, 1, 1);
//SendPack_CCD(2, 3, (uint8_t *)&CCD_BUFF[2], TSL1401_SIZE, 1, 1);
```

八、全局参数设置

能使用的功能是前 4 个按钮的，最后一个暂时没有写。



- 当按下“读取全局参数”按钮，下面代码中第一个 if 语句就会被触发，在这里写上需要回发的参数，主要是 PID 参数和自定义参数。使用方法和前面几个类似，主要是修改对应的数字和变量名字。
- 当按下“紧急停车”按钮，代码中第二个 if 语句就会被触发。请在这里写上能让车停止输出的代码，默认是通过让 PWM 不输出来停止车模。

```
//对于获取下位机参数的响应，写需要回发的变量类型
if(_wholesettings[0].need_Send == 1)
{
    //printf("I will send\n");
    //*****
    //PID参数回发
    //SendPack_PID(SendBackPID, Balance_Stand, ANGLE_CONTROL_P, 0,
    SendPack_PID(SendBackPID, Balance_Speed, SPEED_CONTROL_P, SPEE
    //SendPack_PID(SendBackPID, Balance_Direction, DIRECTION_CONTA
    //*****
    //自定义参数
    //SendPack_Short(SendBackDIY,DIY_Para_1, ANGLE_CONTROL_P, 1, 1
    SendPack_Short(SendBackDIY,DIY_Para_2, 100, 1, 1);
    SendPack_Short(SendBackDIY,DIY_Para_3, 10.2, 1, 1);

    _wholesettings[0].need_Send = 0;
}

//=====
//=====
//紧急停车!!!!!!!!!!!!!!
if(_wholesettings[1].need_Send == 1)
{
    //关闭需要关闭的中断
    //disable_irq(PORTE_IRQn);

    //关闭输出
    ftm_pwm_duty(FTM0, FTM_CH0, 1000);
    ftm_pwm_duty(FTM0, FTM_CH1, 1000);
    ftm_pwm_duty(FTM0, FTM_CH2, 1000);
    ftm_pwm_duty(FTM0, FTM_CH3, 1000);

    SendPack_Echo(9, 1, "ACK");
}
//=====
```

- 当按下“**保存当前运行参数**”，会把“虚拟示波器”和“PID 参数设置”和“自定义算法参数”的值保存到一个配置文件，方便以后使用。（与下位机没关系）
- 当按下“**导出历史数据**”，会把之前保存的配置文件读入到界面，方便使用上一次的参数进行进一步的调试和分析。（与下位机没关系）