HOTINST.COMMUNICATION.Serial的调用方法简易说明

**1.引用类库**

在你的C#工程中引用如下类库：

HOTINST.COMMON.DLL

HOTINST.COMMUNICATION.DLL

HOTINST.COMMUNICATION.Serial.RS232.DLL

HOTINST.COMMUNICATION.Serial.NICAN.DLL

说明：

HOTINST.COMMON.DLL依赖库，必须引用

HOTINST.COMMUNICATION.DLL是工厂模式的接口库，定义了相关接口，必须引用

HOTINST.COMMUNICATION.Serial.RS232.DLL是串行232的实现库，需要才引用

HOTINST.COMMUNICATION.Serial.NICAN.DLL是NICAN的实现，需要才引用

如何引用：

略

**2.配置参数**

可根据需要调用HOTINST.COMMUNICATION.Serial.RS232空间下的CRS232Config类来进行串口配置（NICAN或以后新实现以此类推）。此配置用于串口除端口和波特率外的其它辅助配置，如果不调用配置则自动按默认值处理。注意：配置文件名是默认的且会默认存储到与DLL相同的位置。

调用方法如下（NICAN或以后新实现以此类推）：

//加载配置文件，不存在则创建默认配置文件

CRS232Config objRS232Config = new CRS232Config();

if(!objRS232Config.IsExistsFile())

         objRS232Config.CreateDefaultConfigFile();

objRS232Config.LoadConfigFile();

//按需改变所需配置

objRS232Config.PortParameterList.DataBits = 8;

objRS232Config.PortParameterList.xxx = xxx;

......

//保存配置

SaveConfigFile.SaveConfigFile();

**3.创建操作对象**

ISerialFactory objSerialFactory = CommunicationHelper.CreateSerialFactoryObject();

ISerial objSerial = objSerialFactory.CreateCommObject("PXI-8433");

注："PXI-8433"是板卡或设备的型号，由保存在与HOTINST.COMMUNICATION.DLL 相同位置的COMM\_ProductList.XML文件来配置的，该文件中存在该设备型号则受支持，反之则不受支持。如果需要配置该文件，请使用HOTINST.COMMUNICATION.CProductList类进行操作，如果新增一个设备型号，请确保该设备已有对应的实现类库来支持，以及确保相应的配置项得到正确配置。

**4.为对象配置端口和通讯速率**

objSerial.SetPort(0);

objSerial.SetBaudrate(9600);

注意：这个在每次调用时必须设置，为了方便，建议上层将其存放到上层的配置文件中，实现自动加载和设置。

**5.注册读取数据事件以便接收数据**

objSerial.DataRead += objSerial\_DataRead;

objSerial.DataReadError += objSerial\_DataReadError;

void objSerial\_DataRead(object sender, EventArgs e)

{

         DataReadEventArgs DataReadArgs = (DataReadEventArgs)e;

         byte[] ReadData = DataReadArgs.ByteData;

//也可以使用如下方式

//Stinrg ReadData = DataReadArgs.HexData;

         //处理接收到的数据

}

void objSerial\_DataReadError(object sender, EventArgs e)

{

         objSerial.StopDataRead(); //一般来说正常操作不会有异常，如果检测到异常最好停止读取抛出异常并进行处理更合适。

         DataReadErrorEventArgs ex = (DataReadErrorEventArgs)e;

         MessageBox.Show(ex.ErrorString, "错误", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

**6.打开端口**

objSerial.OpenPort();

**7.接收数据**

objSerial.StartDataRead();

**8.停止接收数据**

objSerial.StopDataRead();

**9.发送数据**

byte[] SendDataBuffer = new byte[3]{ 0x01,0x02,0x03 };

objSerial.WriteData(SendDataBuffer);

**8.关闭端口**

objSerial.ClosePort();