上海格西信息科技有限公司

电子公告牌

版本 0.1

目录

1. 概述

企业中控室与两台数采仪采用 Modbus RTU 协议通信，每台数采仪提供一个 RS232 串口与中控室通信，数采仪作为 Modbus RTU 协议的从站，响应中控室端发送的请求信息。串口设置默认为波特率 9600, 数据位 8, 停止位 1, 无校验。数采仪默认的从机地址为 0x01。

本项目需要监控数采仪两台数采仪的检测参数，如下表所示。寄存器地址从 0000 开始，两个寄存器存放一个 Float 型数值。

序号	数据项目	寄存器地址	单位
1	二氧化硫	0000-0001	mg/m3
2	氮氧化物	0004-0005	mg/m3
3	颗粒物	0006-0007	mg/m3
4	一氧化碳	0008-0009	mg/m3
5	氯化氢	000A-000B	mg/m3
6	氧含量	000C-000D	%
7	烟气流量	000E-000F	m3/h

监控界面如下图所示。

本例子文件位于：<软件安装目录>\Examples\Solutions\BulletinBoard。

文件说明：

✓ BulletinBoard.gpj - 电子公告牌演示项目 - 中文 - 串口版

例子自带仿真器，可以脱离设备仿真运行。

串口版：需要使用串口虚拟软件，如 VSPD 等，虚拟出两对串口（一对为 COM2 和 COM3，一对为 COM4 和 COM5）进行仿真运行。如果虚拟的串口号和例子预定义的串口号不同，可以修改例子串口号，

也可以修改虚拟串口号。

2. 创建项目

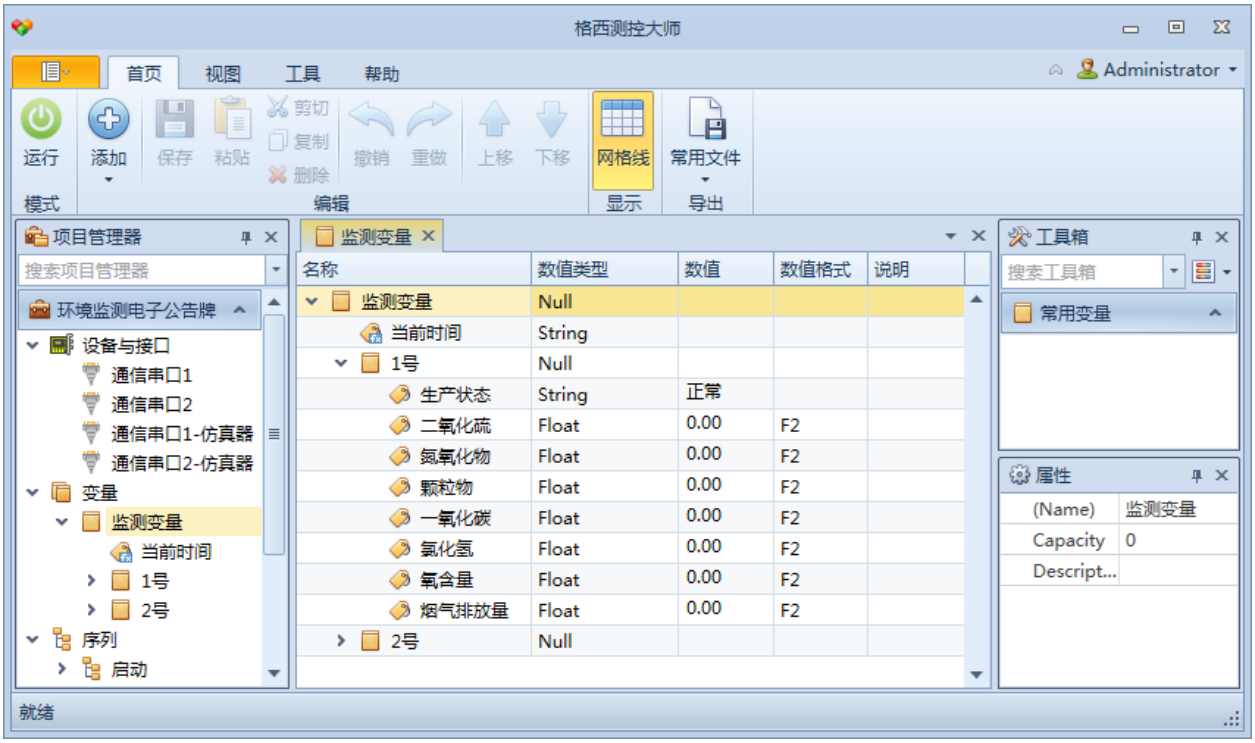
2.1 第 1 步 添加串口设备

本演示项目需要和两台设备连接，分别是 1 号设备和 2 号设备，对应“通信串口 1”和“通信串口 2”；另外，由于本项目仿真了 1 号设备和 2 号设备，分别占用一个串口，对应“通信串口 1-仿真器”和“通信串口 2-仿真器”。

“通信串口 1”和“通信串口 1-仿真器”为 VSPD 虚拟的一对串口 COM2 和 COM3，虚拟交叉线连接，即 COM2 和 COM3 可以互相通信；“通信串口 2”和“通信串口 2-仿真器”同理。

2.2 第 2 步 添加变量

建立变量组，保存采集到的参数数据；同时，变量组中的变量也作为界面显示的桥梁。



2.3 第 3 步 添加序列

本演示项目建立四个序列来实现。

1）“启动”序列：通过脚本实现自动化配置。

```
using System;
using Genesis;
using Genesis.Scripting;
using Genesis.Sequence;
using Genesis.Workbench;
using Genesis.Device;
```

```
public class Step_23BF3840649C42C5A01B8E88E12BB403
{
  public ScriptContext Context { get; set; }

  //
  public Int32 BeginExecute(IStepContext context, IStep step)
  {
    // 打开项目配置文件，以便读取上一次保存的设备参数
    IMemento config = this.Context.OpenProjectConfiguration();
    IMemento devConfig = config.GetChild("Devices");
    // 打开通信接口
    IDeviceSession dev = this.Context.GetDeviceSession("通信串口 1");

    if (devConfig != null)
    {
      IMemento dev1Config = devConfig.GetChild("通信串口 1");
      dev.Address = dev1Config.GetString("Address");
      dev.Parameters = dev1Config.GetString("Parameters");
    }
    dev.Open();

    dev = this.Context.GetDeviceSession("通信串口 2");
    if (devConfig != null)
    {
      IMemento dev2Config = devConfig.GetChild("通信串口 2");
      dev.Address = dev2Config.GetString("Address");
      dev.Parameters = dev2Config.GetString("Parameters");
    }
    dev.Open();

    dev = this.Context.GetDeviceSession("通信串口 1-仿真器");
    if (devConfig != null)
    {
      IMemento dev1simConfig = devConfig.GetChild("通信串口 1-仿真器");
      dev.Address = dev1simConfig.GetString("Address");
      dev.Parameters = dev1simConfig.GetString("Parameters");
    }
    dev.Open();

    dev = this.Context.GetDeviceSession("通信串口 2-仿真器");
    if (devConfig != null)
    {
      IMemento dev2simConfig = devConfig.GetChild("通信串口 2-仿真器");
      dev.Address = dev2simConfig.GetString("Address");
      dev.Parameters = dev2simConfig.GetString("Parameters");
    }
    dev.Open();
```

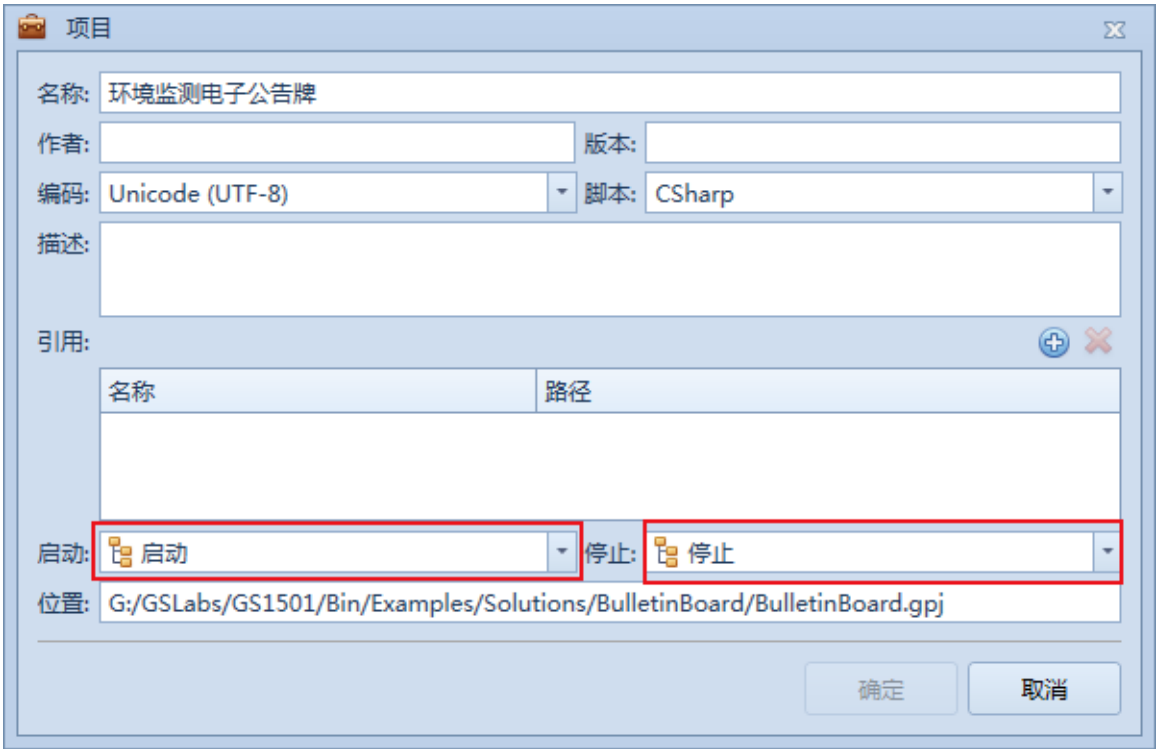
```
    // 运行 采集数据-仿真器序列，仿真 1 号设备和 2 号设备
    this.Context.StartStep("采集数据-仿真器");
    // 运行 采集数据序列
    this.Context.StartStep("采集数据");

    // 关闭工具栏、状态栏、项目管理器等
    this.Context.HideToolBar();
    this.Context.HideStatusBar();
    this.Context.HideEditorHeaders();
    this.Context.CloseAllViews();
    this.Context.CloseAllEditors();
    //this.Context.ShowFullScreen(true);
    // 打开用户界面
    this.Context.OpenSchema("公告界面");
    return 0;
  }
}
```

2）“停止”序列：通过脚本实现程序关闭后自动保存串口设置到配置文件。

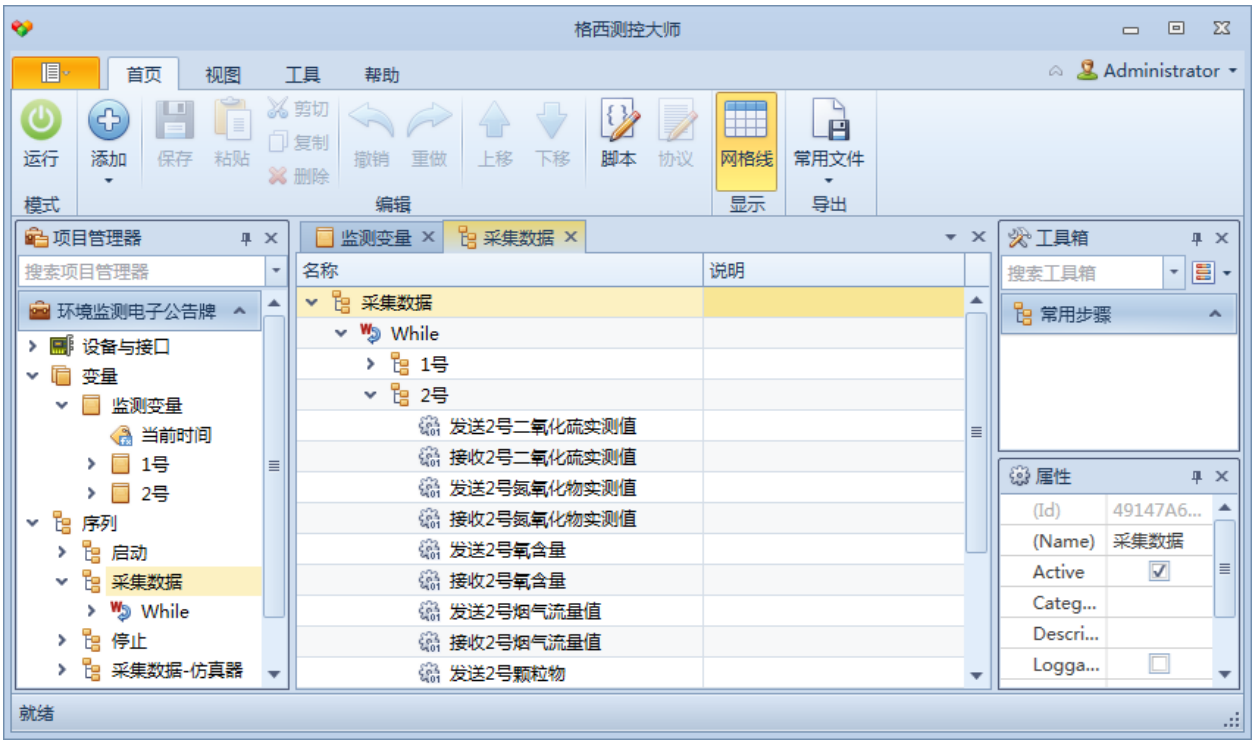
```
using System;
using Genesis;
using Genesis.Scripting;
using Genesis.Sequence;
using Genesis.Workbench;
using Genesis.Device;

public class Step_AE021E1E27F24BA4BED5DAE13B040900
{
  public ScriptContext Context { get; set; }

  //
  public Int32 BeginExecute(IStepContext context, IStep step)
  {
    // 创建和保存项目配置文件
    IMemento config = this.Context.CreateProjectConfiguration();
    IMemento devConfig = config.CreateChild("Devices");
    // 保存通信接口信息
    IMemento dev1Config = devConfig.CreateChild("通信串口 1");
    IDeviceSession dev = this.Context.GetDeviceSession("通信串口 1");
    dev1Config.PutString("Address",dev.Address);
    dev1Config.PutString("Parameters",dev.Parameters);

    IMemento dev2Config = devConfig.CreateChild("通信串口 2");
    dev = this.Context.GetDeviceSession("通信串口 2");
    dev2Config.PutString("Address",dev.Address);
    dev2Config.PutString("Parameters",dev.Parameters);
```

```
    IMemento dev1simConfig = devConfig.CreateChild("通信串口 1-仿真器");
    dev = this.Context.GetDeviceSession("通信串口 1-仿真器");
    dev1simConfig.PutString("Address",dev.Address);
    dev1simConfig.PutString("Parameters",dev.Parameters);

    IMemento dev2simConfig = devConfig.CreateChild("通信串口 2-仿真器");
    dev = this.Context.GetDeviceSession("通信串口 2-仿真器");
    dev2simConfig.PutString("Address",dev.Address);
    dev2simConfig.PutString("Parameters",dev.Parameters);
    // 保存文件
    this.Context.SaveProjectConfiguration(config);
    return 0;
  }
}
```

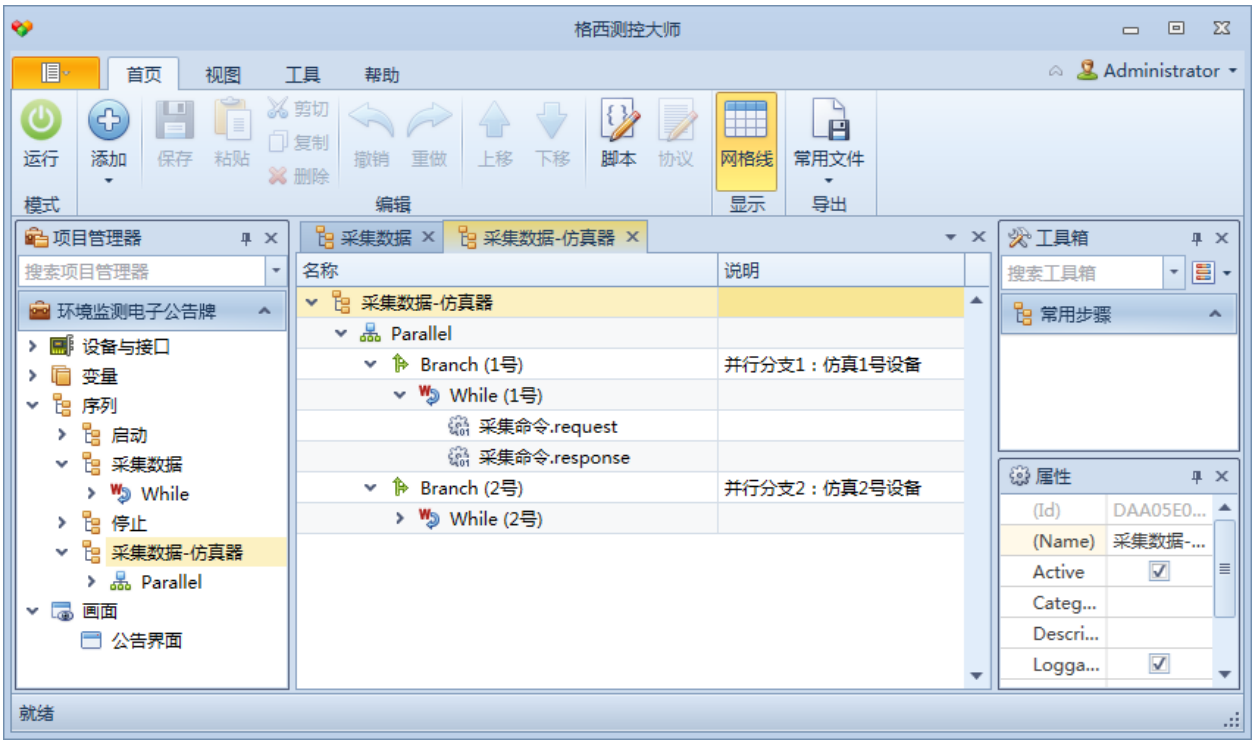
启动和停止序列的执行在项目属性框中设置。



3）“采集数据”序列：通过 Protocol 型步骤实现 1 号设备和 2 号设备的逐个参数采集，并设置到相应的变量中，然后自动显示在用户界面上。

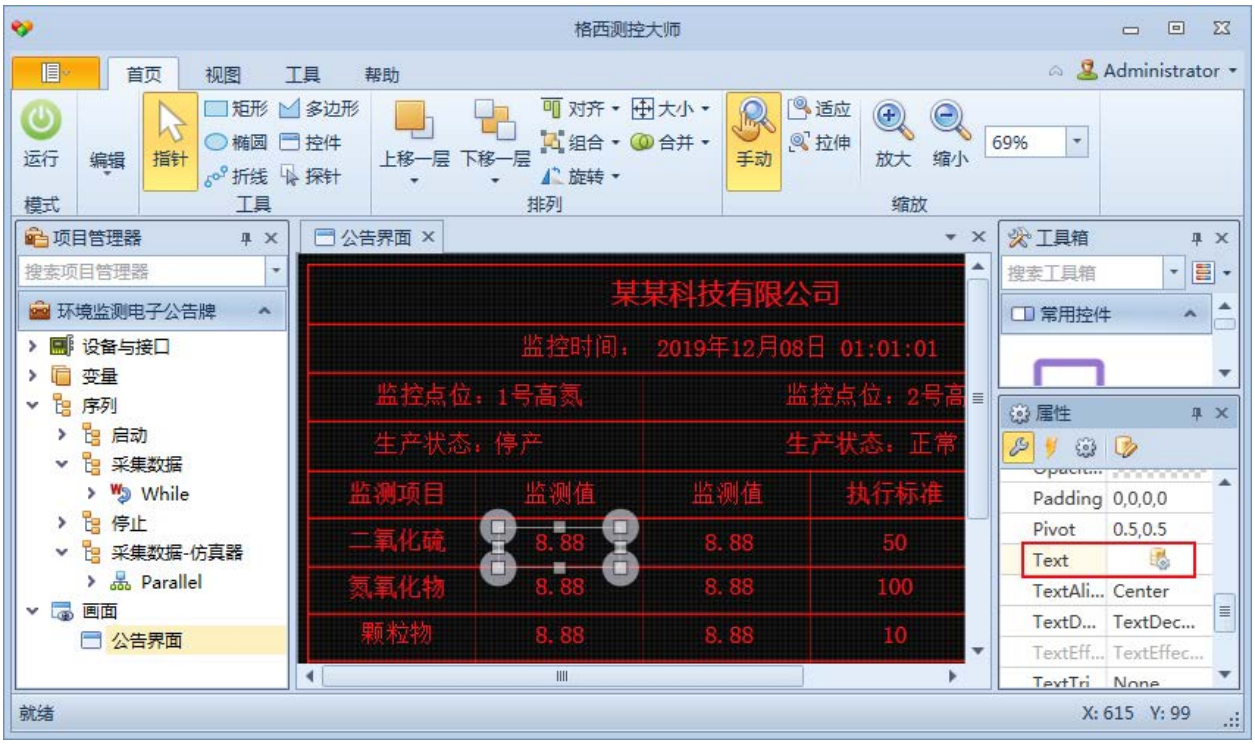


4）“采集数据-仿真器”序列：通过 Protocol 型步骤实现 1 号设备和 2 号设备的命令仿真。

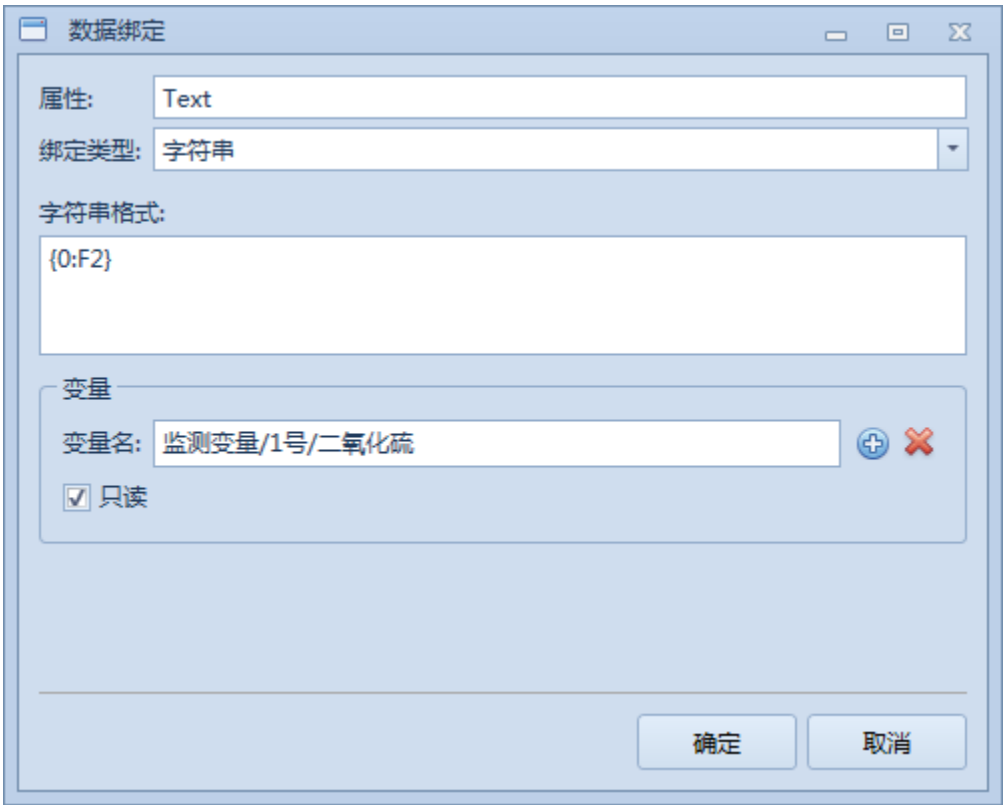


2.4 第 4 步 添加界面

本演示项目的画面采用 Border 控件和 TextBlock 控件实现。用于显示采集参数的 TextBlock，通过数据绑定的方式把对应的变量绑定到 Text 属性上，这样当变量的值变化时，就能显示在控件上。



双击 Text 属性，弹出“数据绑定”对话框，如下图所示。



3. 运行项目

3.1 打开并运行项目

从<软件安装目录>\Examples\Solutions\BulletinBoard 目录中，打开 BulletinBoard.gpj 串口版项目文件。点击工具栏的“运行”按钮，切换到运行模式。由于项目属性中“启动”属性已经设置了启动序列为序列列表中的“启动”序列，故自动执行“启动”序列中的脚本，脚本依次执行：

1） 打开项目配置文件，读取上一次保存的设备参数并设置设备参数
2） 运行“采集数据-仿真器”序列，仿真 1 号设备和 2 号设备
3） 运行“采集数据”序列，开始采集
4） 关闭工具栏、状态栏、项目管理器等界面组件，实际使用也可以直接打开全屏模式
5） 打开公告牌界面

如果要返回普通操作界面，通过快捷键“F9”重置窗口，重置后如下图所示。