目　　录

官方网站：http://www.bmpj.net

2.服务实例的配置参数说明

2.1 概述

SuperIO(SIO)定位在 PC 终端（上位机）应用，它只有一个服务实例，配置参数是全局属性。但是，ServerSuperIO（SSIO）与 SuperIO(SIO)定位不一样，SSIO 定位在服务器端，不管是串口通讯模式，还是网络通讯模式，都支持多服务实例，所以每个服务实例都有自己的配置参数，全部配置参数的定义在 ServerConfig.cs 文件中。

如下图示意：

2.2 配置参数说明

```
[Category("1.全局"),
  DisplayName("ServerSession"),
  Description("标识服务的唯 ID，一般为 Guid"),
  DefaultValue(""),
  ReadOnly(true)]
  public string ServerSession { get; set; }

[Category("1.全局"),
DisplayName("ServerName"),
Description("标识服务的标题名称"),
DefaultValue("")]
```

```
public string ServerName { get; set; }

[Category("1.全局"),
DisplayName("DeliveryMode"),
Description("接收数据后的分布策略，包括：按设备 IP 分发（DeviceIP）、
按设备编码分发（DeviceCode）"),
DefaultValue(DeliveryMode.DeviceIP)]
public DeliveryMode DeliveryMode { get; set; }

[Category("1.全局"),
  DisplayName("ControlMode"),
  Description("调度设备驱动和 IO 实例的策略，包括:循环模式（Loop）、并
发模式（Parallel）、自主模式（Self）和单例模式（Singleton）"),
  DefaultValue(ControlMode.Loop)]
public ControlMode ControlMode { get; set; }

[Category("1.全局"),
  DisplayName("StartReceiveDataFliter"),
  Description("标识接收数据后是否按协议过滤器的规划过滤数据,不启用则
直接返回数据"),
  DefaultValue(false)]
public bool StartReceiveDataFliter { get; set; }

[Category("1.全局"),
DisplayName("StartCheckPackageLength"),
Description("标识是否检测数据长度，如果开启，那么会调用协议驱动的
GetPackageLength 接口，直到接收返回的数据长度的数据"),
DefaultValue(false)]
public bool StartCheckPackageLength { get; set; }
#endregion

#region 串口
[Category("2.串口"),
  DisplayName("ComReadBufferSize"),
  Description("设置一次接收数据的字节数组最大值"),
  DefaultValue(1024)]
public int ComReadBufferSize { get; set; }

[Category("2.串口"),
DisplayName("ComWriteBufferSize"),
Description("设置一次发送数据的字节数组最大值"),
DefaultValue(1024)]
public int ComWriteBufferSize { get; set; }
```

```
[Category("2.串口"),
DisplayName("ComReadTimeout"),
Description("设置一次读取数据的超时时间"),
DefaultValue(1000)]
public int ComReadTimeout { get; set; }

[Category("2.串口"),
DisplayName("ComWriteTimeout"),
Description("设置一次发送数据的超时时间"),
DefaultValue(1000)]
public int ComWriteTimeout { get; set; }

[Category("2.串口"),
DisplayName("ComLoopInterval"),
Description("轮询模式下，发送和接收数据中间的等待时间，串口通讯不支持
其他控制模式"),
DefaultValue(1000)]
public int ComLoopInterval { get; set; }
#endregion

#region 网络
[Category("3.网络"),
  DisplayName("NetReceiveBufferSize"),
  Description("设置一次接收数据的字节数组最大值"),
  DefaultValue(1024)]
public int NetReceiveBufferSize { get; set; }

[Category("3.网络"),
DisplayName("NetSendBufferSize"),
Description("设置一次发送数据的字节数组最大值"),
DefaultValue(1024)]
public int NetSendBufferSize { get; set; }

[Category("3.网络"),
DisplayName("NetReceiveTimeout"),
Description("设置一次读取数据的超时时间"),
DefaultValue(1000)]
public int NetReceiveTimeout { get; set; }

[Category("3.网络"),
  DisplayName("NetSendTimeout"),
  Description("设置一次发送数据的超时时间"),
  DefaultValue(1000)]
public int NetSendTimeout { get; set; }
```

```
[Category("3.网络"),
DisplayName("NetLoopInterval"),
Description("轮询模式下，发送和接收数据中间的等待时间"),
DefaultValue(1000)]
public int NetLoopInterval { get; set; }

[Category("3.网络"),
DisplayName("MaxConnects"),
Description("允许客户端最大的连接数，超取最大值，自动关闭远程连接"),
DefaultValue(1000)]
public int MaxConnects { get; set; }

[Category("3.网络"),
DisplayName("KeepAlive"),
Description("检测死连接、半连接的一种机制"),
DefaultValue(5000)]
public uint KeepAlive { get; set; }

[Category("3.网络"),
DisplayName("ListenPort"),
Description("侦听接收数据的端口"),
DefaultValue(6699)]
public int ListenPort { get; set; }

[Category("3.网络"),
DisplayName("BackLog"),
Description("定队列中最多可容纳的等待接受的传入连接数"),
DefaultValue(1000)]
public int BackLog { get; set; }

[Category("3.网络"),
DisplayName("CheckSameSocketSession"),
Description("对一个固定的设备，只允许有一个有效连接，重复 IP 多次连接，
将断开之前的连接"),
DefaultValue(true)]
public bool CheckSameSocketSession { get; set; }

[Category("3.网络"),
DisplayName("SocketMode"),
Description("标识设备是 TcpServer、TcpClient 模式，如果标识 TcpClient
模式，会主动连接远程 IP 和端口"),
DefaultValue(SocketMode.Tcp)]
public SocketMode SocketMode { get; set; }
```

```
[Category("3.网络"),
DisplayName("ClearSocketSession"),
Description("标识是否清理连接,如果一个连接在一定时间范围内没有接收到数据，将主动断开连接"),
DefaultValue(false)]
public bool ClearSocketSession { get; set; }

[Category("3.网络"),
DisplayName("ClearSocketSessionInterval"),
Description("如果标识清理连接，那么在此标识清理连接间隔时间"),
DefaultValue(10)]
public int ClearSocketSessionInterval { get; set; }

[Category("3.网络"),
DisplayName("ClearSocketSessionTimeOut"),
Description("如果标识清理连接,那么在此标识多长时间没有接收到数据进行清理"),
DefaultValue(30)]
public int ClearSocketSessionTimeOut { get; set; }
```

2.3 常用配置参数说明

常用的配置参数包括：通讯参数类、控制参数类、以及一些高级的应用参数。

代码如下：

```
IServer server = new ServerFactory().CreateServer(new ServerConfig()
{
      ServerName = "服务 1",      //服务实例的名称
      ComReadTimeout = 1000, //串口读数据超时
      ComWriteTimeout = 1000, //串口发送数据超时
      NetReceiveTimeout = 1000, //网络接收数据超时
      NetSendTimeout = 1000,//网络发送数据超时
      ControlMode = ControlMode.Parallel,//控制模式
      SocketMode = SocketMode.Tcp, //网络通讯是 TCP 模式还是 UDP 模式
      StartReceiveDataFliter = false,//是否开启接收数据过滤器，后面重要介绍
      ClearSocketSession = false,//是否检测网络实例的有效性，后面重要介绍
      StartCheckPackageLength = false //是否检测包长度，后面重要介绍
});
```

ControlMode 参数是 SSIO 结合现实应用场景的控制模式，主要用于调用设

备的发送和接收数据的调度方式。请参见：《连载 | 物联网框架 ServerSuperIO 教程》1.4 种通讯模式机制。

2.4 配置工具

二次开发者，可以通过 ServerSuperIO.Tool 项目来配置服务实例、设备驱动和服务实例的参数。如下图：