SIEMENS西门子6SN1118-0NH01-0AA1模块代理商

产品名称	SIEMENS西门子6SN1118-0NH01-0AA1模块代理商
公司名称	湖南西控自动化设备有限公司
价格	.00/件
规格参数	西门子:西门子授权代理商 备件:核心供货商 德国:现货
公司地址	中国（湖南）自由贸易试验区长沙片区开元东路1306号开阳智能制造产业园（一期）4#栋301
联系电话	17838383235 17838383235

产品详情

一个连接西门子PLC设备的.net库，搞自动化的有福了！

【导读】推荐一个连接西门子PLC的.net库，包括使用方法以及代码。

前言

PLC在工业自动化领域是常用的控制器，一般在和上位机界面通讯时，经常使用组态软件。

以西门子PLC为例，上位机可以使用西门子的WINCC。但是当面对需求比较多样化的需求时，WINCC难以胜任。而且作为量产的标准化产品，WINCC授权的费用一笔不小的成本。

S7.Net.dll 是应用在
.NET平台上和西门子PLC通讯的一个动态库。大家可以到GitHub中直接去下载，或者直接搜索“ S7.Net .dll ”也可以找到下载的链接。文章后也会给出链接。

（一）S7Net动态库说明

目前该动态库支持的西门子PLC类型有S7-200,S7-300,S7-400,S7-1200和S7-1500，基本涵盖了西门子所有的PLC产品。该库可以直接读写PLC中的全局DB块，作为通讯的接口。

下面对需要的一些函数以及数据类型做简单说明。构造函数：cpu 为枚举类型，代表PLC类型。ip 为PLC地址，需要和PLC组态的地址一致，同时，和通讯的PC IP地址应在同一局域网段。rack 为导轨号，slot 为插槽号，均可在博途硬件组态处获得。

如下图所示，ip=“ 192.168.0.5 ”。

如下图所示，rack为0，slot为1。

public Plc(CpuType cpu, string ip, short rack, short slot); public enum CpuType {
S7200 = 0,　　S7300 = 10,　　S7400 = 20,
S71200 = 30,　　S71500 = 40 }

获取是否连接成功：

public bool IsConnected { get; }

连接PLC：

public void Open(); public Task OpenAsync();

该类库提供了两个用于PLC连接的函数，区别在于第二个为异步连接，在请求连接的过程中，不会导致线程阻塞。我个人比较喜欢第二种方式。

按位写操作：参数db代表访问的DB块编号，如下图所示UISendInt编号为3，UIReadInt编号为4。

startByteAdr是在数据块内以字节为单位的起始地址，可以传入0。bitAdr 为需要操作的位的偏移地址（以startByteAdr为基准），value 为写入的值。dataType为枚举类型，我们操作的是数据块，传入DataBlock。

public void WriteBit(DataType dataType, int db, int startByteAdr, int bitAdr, bool value);
public enum DataType {　Counter = 28,　Timer = 29,
Input = 129,　Output = 130,　Memory = 131,　DataBlock = 132
}

按字节读：count代表读的字节数量。在进行按字节读写时，对于8位的单字节变量来说没有任何问题。但是对于多字节比如在PLC中16位I的INT类型，需要注意大小端问题。在西门子PLC中以大端模式存储数据，但是在Intel的X86架构的PC上，却是以小端模式。所以，在从PLC读取一个INT类型的变量，需要将读取回来的字节数组高低字节调换，后面C#程序中会有详细的实现代码。

public byte[] ReadBytes(DataType dataType, int db, int startByteAdr, int count);

按字节写：value代表将要传输的数据转换成的字节数组，如果要操作一个INT变量，需要提前将高低字节调换，后面C#程序中会有详细的实现代码。

public void WriteBytes(DataType dataType, int db, int startByteAdr, byte[] value);

（二）PLC设置

实现外部程序可以访问操作DB块，需要对PLC部分做一些设置。首先对于要读取或者访问的DB块，要将优化的块访问取消，默认为勾选的，选中块右击选择属性。

同时还要在硬件组态中，设置防护与安全中的连接机制。将“ 允许来自远程对象的PUT/GET访问 ”勾选。如下图：

（三）C#程序

在C#上位机软件中，对通讯类库的函数做一个简单的包装，以方便我们使用。首先新建一个类S7，在项目中需要引用S7.NET，并在该类中包含命名空间。

using S7.Net;

定义一个PLC类型变量PLC，并定义该类的构造函数和析构函数：

Plc plc; public S7(S7DataType.CpuType cpuType,string ip,Int16 rack,Int16 slot) {
plc = new Plc((CpuType)cpuType,ip, rack,slot); } ~S7() { this.plc.Close(); }

定义连接函数以及关闭函数：

public void OpenAsync() { this.plc.OpenAsync(); } public void Close() {
this.plc.Close(); }

定义读写函数：之所以加lock,是因为在应用中数据通讯很有可能在不同的线程去操作，比如单独开一个线程定时从PLC更新数据。

public byte[] PlcReadBytes(int db, int startByteAdr=0, int count=1) { try
{ lock (this) {
return this.plc.ReadBytes(DataType.DataBlock, db, startByteAdr, count); }
} catch { return new byte[2]; }
} public void PlcWriteBytes(int db, byte[] value, int startByteAdr = 0) {
lock (this) {
this.plc.WriteBytes(DataType.DataBlock, db, startByteAdr, value); } }

```
public void PlcWriteBit(int db, int bitAdr, bool value, S7DataType.DataType dataType =
S7DataType.DataType.DataBlock) {     int temp1 = bitAdr / 8;
int temp2 = bitAdr % 8;     lock (this)     {
this.plc.WriteBit((DataType)dataType, db, temp1, temp2, value);     } }
```

定义字节和Int16类型之间的转换函数（需要进行高低字节转换）：

```
public static byte[] Int16ToBytes(Int16 data) {      byte[] temp = new byte[2];
temp[0] =(byte) (data >> 8);      temp[1] = (byte)(data);      return temp;
} public static Int16 BytesToInt16(byte[] bytes) {      Int16 temp;
temp =(Int16)( bytes[0] << 8 | bytes[1]);      return temp;  }
```