

PLC-Link.NET for Omron – Robust, high-speed PLC integration

Overview

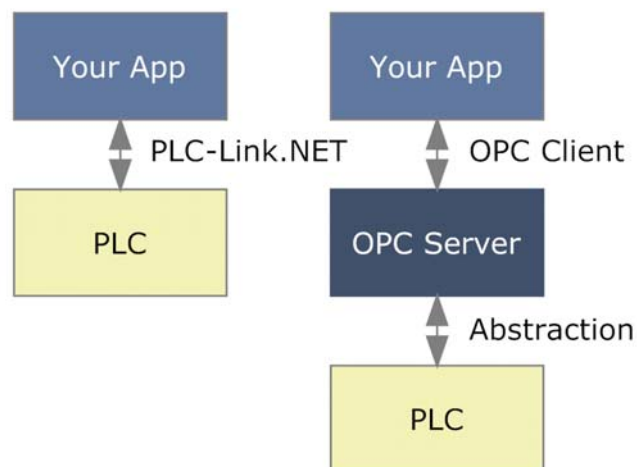
When it comes to the integration of automation systems, time is of the essence, whether it's the cycle time of the process or the time required to build it. PLC-Link.NET has been developed from the ground up for performance and ease of integration.

Key Benefits

- Field-proven in applications requiring high performance and reliability
- Easy to use and maintain
- Designed specifically for .NET
- Tightly integrates your application with the PLC
- Competitively priced

Applications

- Control-centric applications, where two-way communication is a must
- Applications that require fast response times from the PLC
- Applications that require predictable control over how PLC memory is accessed
- Applications that don't require the complicated features provided by an OPC server



Why PLC-Link.NET?

- Better performance than general-purpose packages, such as OPC
- All communication takes place in your own application—no outside server processes required
- Easier to configure and maintain
- Lower per-unit costs compared with OPC

Summary

If you are integrating an Omron PLC and could benefit from higher performance, reduced costs and improved ease of use, please contact us for additional information. Accelerate your project with our many years of Omron expertise.

**Positronics
Incorporated**

"a premier software solution
provider for robotic, motion and
machine control applications."

(925) 931-0211

www.posincorp.com

sales@posincorp.com



Technical Features

- High-speed communication over Ethernet using FINS UDP
- Native .NET classes designed to feel like an extension of the framework
- Block reads and writes of arbitrary PLC memory locations
- Efficient random-access reads for sparsely distributed data
- Robust error detection and tolerance for noisy network connections
- Full support for multiple threading
- Supports both synchronous and asynchronous usage

Connect to a PLC

```
FinsUdpClient client = new FinsUdpClient();
client.Connect("192.168.1.20");
```

Easily access memory for quick prototyping

```
int D500value = client.ReadInt16(OmronAddress.DM(500)); //Read signed INT at D500
client.WriteBoolean(OmronAddress.CIO(100, 5), true); //SET 100.5
client.WriteBCD16(OmronAddress.Parse("800"), 1234); //Write 0x1234 to 800
```

Efficient block reads of complex data

```
OmronDecoder decoder = client.Read(OmronAddress.CIO(700), 6); //6 words at 700
int header = decoder.ReadInt32(); //First two words as a signed DINT
string text = decoder.ReadString(8); //Next 8 bytes (702-705) as an ASCII string
```

Efficient random-access reading

```
//Read 0.00, W0.04, and 1.03 in a single message
decoder = client.MultipleRead(OmronAddress.CIO(0, 0),
                             OmronAddress.WR(0, 4),
                             OmronAddress.CIO(1, 3));
```

```
//Retrieve them one at a time
bool flag000 = decoder.ReadBoolean();
bool flagW004 = decoder.ReadBoolean();
bool flag103 = decoder.ReadBoolean();
```

Efficient block writes of complex data

```
OmronEncoder encoder = new OmronEncoder();
encoder.WriteBCD16(9876); //Write 0x9876 in first two bytes
encoder.WriteInt32(-1).WriteString("hello", 10); //Write -1 and a 10 byte string
client.Write(OmronAddress.DM(6000), encoder); //Write data to address D6000
```

