AB Rockwell ControlLogix Connector

ControlLogix 5000, FlexLogic and CompactLogix

- Name: **ControlLogix**
- Version 1.9.1.9
- Protocol: CIP
- Interface: TCP/IP
- Runtime: .NET 2.0 (Multiplatform)
- Configuration:
  - Devices / Protocols
  - UNS / TagProviders

Configuration Workflow
Channels Configuration
Nodes Configuration
Points Configuration
Performance Optimization
Troubleshoot
Driver Revision History

Configuration Workflow

This document has only the specific information about the protocol connection settings and its address syntax.
Go To Data Connection models, for information on the configuration options.

- UNS LocalTags + Devices

1. Create local tags
2. On Devices, create the Channel, Node, and map the Tag to a an address
3. Alternatively with the Import Wizard for automated creation.

- UNS LocalTags + Devices

1. Create local tags
2. On Devices, create the Channel, Node, and map the Tag to a an address
3. Alternatively with the Import Wizard for automated creation.

This document has only information about the device connection settings and its address syntax.

For further Information, see:

- Devices Module
- Devices Module Reference

Channels Configuration

Protocol Options

Model: Set the PLC model. It can be:

- ***Others:*** For all models except 1756-L8X and Micro850.
- ***1756-L8X:*** For Allen-Bradley model: 1756-L8X, 5069-L330ERS, CompactLogix 5380 and 5069-L310ERS2 5380 GuardLogix.
- ***Micro850:*** For the Micro850 series.

MaxStringItemsPerBlock: Define the maximum string item count to each block.

For example, if there are 12 string items configured and the MaxStringItemsPerBlock is 3, the result will be 4 blocks to read string data.

WaitingBetweenMsgs: Sets the delay time between consecutive messages sent to the device, helping manage communication load.

StringLegacy: Configures compatibility with legacy string formats used in older systems.

Nodes Configuration

Primary Station

Stations syntax: <IP > ; <Port > ; <Slot>

Where :

<**IP**> = IP address of the slave device in the network

< **Port** > = TCP port where the slave device is listening (default is 44818)

<**Slot**> = Slot is the Slot number where the CPU is connected.

Example: 192.168.1.101; 44818; 0

Points Configuration

The syntax for the ControlLogix communication points is: <Type> : <DeviceTagName>:<CIPLIttleEndian>

- **Type:** Type is data type of the Tag in the PLC.
- **DeviceTagName**: name of the Tag in the PLC.
- **CIPLIttleEndian:** controls whether the driver forces CIP little-endian byte ordering for the data exchanged on a specific Point.

The valid type values are:

ControlLogix Address Types				
Type	**Read**	**Write**	**Size**	**Range of Value**
BOOL	**Yes**	**Yes**	1 bit	0 or 1
SINT	**Yes**	**Yes**	1 byte or 8 bits	-128 to 127
INT	**Yes**	**Yes**	2 bytes or 16 bits	-32768 to 32767
DINT	**Yes**	**Yes**	4 bytes or 32 bits	-2,147,483,648 to 2,147,483,647
REAL	**Yes**	**Yes**	4 bytes or 32 bits IEEE Floating point	$-9.99x10^{37}$ to $9.99x10^{37}$
STRING	**Yes**	**Yes**	n bytes	---

The following Table has possible formats for the **DeviceTagName**

ControlLogix Tag Name formats		
Format	**Syntax**	**Comments**
Standard	<Device Tag Name>	
Array Element	<Device Array Tag name>[dim 1, dim2, dim 3]	Dimension Range = 1 to 3
UserDefinedType (UDT)	<Decice Main Tag Name>.<Type Member>	
UserDefinedType (UDT) Array Element	<Device Main Tag Name>.<Type Member>[dim 1, dim2, dim 3]	Dimension Range = 1 to 3

CIPLittleEndian and **ByteSwap**:

When CIP Little Endian is enabled, the byte order is handled automatically by the driver and any byte-level modifiers such as Byte Swap configured in the Modifiers column will have no effect.
To use ByteSwap modifiers on a Point, you must first disable CIP Little Endian for that Point.
After disabling CIPLittleEndian, the ByteSwap can be configured per Point using the Modifiers column in the Points configuration.

Accessing Bit from SINT or INT or DINT ControlLogix datatype

Method 1: Using the *Modifiers* column.

Use the Modifiers column in Device -> Points to specify the Bit to access.

Example of Device Point Configuration to access Bit 3 from INT

TagName	Node	Address	DataType	AccessType	Modifiers
Integer	Node1	INT:PLC_INTEGER	Native	ReadWrite	Bit = 3

Method 2: Using the Tag property.

In the Device the whole word into a tag and use the Tag property to access the specify Bit.

Example of Device Point Configuration to access whole word

TagName	Node	Address	DataType	AccessType
Integer	Node1	INT:PLC_INTEGER	Native	ReadWrite

Program-Scope Tags

By default, the address syntax described above resolves against **Controller-scope (Global) Tags**. To address a tag declared inside a specific Program, prefix the tag name with `Program:<ProgramName>.<TagName>`.

Syntax: `<Type>:Program:<ProgramName>.<TagName>`

Example: `DINT:Program:MainProgram.Counter` — reads the DINT tag `Counter` declared in the local-scope tag table of the program named `MainProgram`.

The same prefix supports UDT members and array elements: `REAL:Program:LineA.PumpUDT.Pressure`, `INT:Program:Mixer.SetpointArray[3]`.

> ⚠ The `Program:<ProgramName>` prefix consumes additional bytes in every Multi-Item Request packet. For high-traffic tags, promoting them to Controller-scope (Global) Tags keeps the protocol packet shorter and improves throughput — see **Performance Optimization Use Global Tags Only** below.

Syntax to access Tag Bit property in any place of project

Tag.<*TagName*>.Bit<*Bit Number*>

E.g.: Tag.Interger.Bit3

Examples

ControlLogix Address Configurations Examples				
TagName	**Node**	**Address**	**DataType**	**AccessType**
Digital	Node1	BOOL:PLC_BOOLEAN	Native	ReadWrite
Digital[2]	Node1	BOOL:BOOLEAN_ARRAY[2]	Native	ReadWrite
DigitalUDT	Node1	BOOL:MAINTAG.PLC_BOOLEAN	Native	ReadWrite
DigitalUDT[4]	Node1	BOOL:MAINTAG.BOOLEAN_ARRAY[4]	Native	ReadWrite
Word	Node1	SINT:PLC_SINTEGER	Native	ReadWrite
Word[7]	Node1	SINT:SINTEGER_ARRAY[7]	Native	ReadWrite
WordUDT	Node1	SINT:MAINTAG.PLC_SINTEGER	Native	ReadWrite
WordUDT[8]	Node1	SINT:MAINTAG.SINTEGER_ARRAY[8]	Native	ReadWrite
Integer	Node1	INT:PLC_INTEGER	Native	ReadWrite
Interger[3]	Node1	INT:INTEGER_ARRAY[3]	Native	ReadWrite
IntegerUDT	Node1	INT:MAINTAG.PLC_INTEGER	Native	ReadWrite
IntegerUDT[10]	Node1	INT:MAINTAG.INTEGER_ARRAY[10]	Native	ReadWrite
Double	Node1	DINT:PLC_DINTEGER	Native	ReadWrite
Double[14]	Node1	DINT:DINTEGER_ARRAY[14]	Native	ReadWrite
DoubleUDT	Node1	DINT:MAINTAG.PLC_DINTEGER	Native	ReadWrite
DoubleUDT[12]	Node1	DINT:MAINTAG.DINTEGER_ARRAY[12]	Native	ReadWrite
Real	Node1	REAL:PLC_REAL	Native	ReadWrite
RealArray[5]	Node1	REAL:REAL_ARRAY[5]	Native	ReadWrite
RealUDT	Node1	REAL:MAINTAG.PLC_REAL	Native	ReadWrite

RealUDT[34]	Node1	REAL:MAINTAG.REAL_ARRAY[34]	Native	ReadWrite
Text	Node1	STRING:PLC_STRING	Native	ReadWrite
TextArray[26]	Node1	STRING:STRING_ARRAY[26]	Native	ReadWrite
TextUDT	Node1	STRING:MAINTAG.PLC_STRING	Native	ReadWrite
TextUDT[21]	Node1	STRING:MAINTAG.STRING_ARRAY[21]	Native	ReadWrite

---

Performance Optimization

Overview

When you communicate with a ControlLogix PLC, the protocol is not as efficient as the older PLC 5 style N7:0 addressing. This is the price you pay for the convenience of using tag names in a PLC.

With the ControlLogix PLC, you actually have to give it the complete tag name of each tag that you want. The PLC has to find that tag in its database and return the data to you.

When ControlLogix first came out, you could only get one item in each packet, which meant performance was quite slow. Years later, Rockwell added what they call the CIP (Controller Interface Protocol) Multi-Item Request Packet.

ControlLogix Multi-Item Request Packet sizes are about 500 bytes and can contain the names of multiple PLC tag names that you would like to read. Every character in a tag name takes up about 1 byte.

Some of the 500 bytes are needed for the PLC command and general communication overhead. The point remains the same though: there are less than 500 bytes available to fill tag names into a request packet.

Maximizing System Throughput

As with any programmable controller, there are a variety of ways to enhance the performance and system communications.

In this document, we will discuss the following items:

- Set the PLC CPU time slice to 40%-50%
- Use Array Tags
- Use Global Tags only
- Simultaneous Connections

Set the PLC CPU time slice to 40%-50%

One of the first things you should check in your system is the CPU time slice setting in your ControlLogix PLC. This is set using your RS Logix 5000 programming software. This value represents the percentage of available CPU time that is dedicated to communication activities by the PLC CPU.

It should be noted that the time slice settings for the CPU communications default is 10%. For every 10 ms program scan that occurs, the controller spends 1 ms processing driver requests.

If the communication tasks are high priority, the time slice should be set at or above 30%. Set the slice from 10% to 40% to achieve the best balance of communication, performance, and CPU usage. Values greater than 50% start to reach diminishing returns.

Use Array Tags

The best way to communicate with ControlLogix is through the use of Arrays and short PLC tag addresses. Arrays may have a long tag address name in the PLC. If there are numerous tags in the array, great gains in performance can be achieved due to the ability of a single request for all the tags within the 500-byte packet size.

One hundred tags can be read or written to in a single command. For example: If the Array size is 100 (e.g.: MyTag is an array MyTag[0] to MyTag [99]), under 200ms would be reasonable yet still dependent on the number of other requests required to be met by the system and demands on ControlLogix.

Boolean arrays are treated differently: in protocol, a Boolean array is a 32-bit array. Thus, requesting element 0 is requesting bits 0 through 31.

Use Global Tags Only

In PLCs, the use of short tag names for tags other than arrays is also of great benefit. This is because the TOP Server packs the PLC tag addresses into the Multi-Item Request packet sent to ControlLogix. The 500-byte limit is what makes the shortness of the tag addresses so critical.

Creating all required PC communication tags under the Global file is one way of shortening the names because Global tags require the least amount of space in the Multi-Item Request Packet.

Local (e.g.: Program) tags may sound nice, but in the ControlLogix to get at a program tag (vs. a Global) we also have to put the text: **Program: ProgramName** in the packet. Plus, we have to add the tag name. You can see how 500 bytes can go fast.

Since we can import L5K file tag descriptions, it is not hard to then change the Server tag names to something more descriptive which would make HMI project creation easier. The key is that the ControlLogix PLC tag name be as short as possible, not the tag name used in the SCADA database.

Simultaneous Connections

In general, the limit on the number of Ethernet connections that can be made to a ControlLogix PLC is high. The default ControlLogix driver opens one TCP socket or connection to the PLC's Ethernet module for each channel that is defined in the SCADA. However, in the ControlLogix **Channel Settings** Column, you can change the number of the NodeConnections and Simultaneous Connections (default 1 and 1 respectively).

For example: If a user created a ControlLogix Channel with 15 node connections and 15 simultaneous connections, 15 TCP socket connections would be made. It means that you have 15 messages going to the PLC at the same time. It is 15 time faster than only one connection. In theory, if the PLC's Ethernet module had a limit of 64 simultaneous TCP sockets, users could connect 64 simultaneous connection at the same time to the PLC.

As more connections are made to a controller, communication performance will begin to degrade based on the amount of data being acquired and the rate of acquisition. To refine the ratio of CPU time spent on processing and communications, refer to the PLC's System Overhead Time Slice (SOTS) parameter description.

In summary, this can greatly increase the performance of your communication, but you need to balance it with the PLC's Time Slice.

Troubleshoot

The status of the driver execution can be observed through the platform built-in diagnostic tools. Refer to the Using Diagnostic Tools topic in Devices Communication page.

Connection Error Codes

Connection errors		
Error Code	**Description**	**Possible Solution**
0	Success	• None
-100	Error Sending Message	• Turn PLC on • Plug the PLC Ethernet cable • Check configured IP Address field in **Device Node** • Ping PLC using prompt command
-101	Error Sending and Waiting Message	
-102 … -105	Error creating TCP/IP connection	
-106	Error Receiving Message	
-112	Timeout Start Message	• Turn PLC on • Plug the PLC Ethernet cable • Ping PLC using prompt command • Check configured IP Address field in **Device Node** • Increase the driver timeout field in **Device Channel**
-113	Timeout between Treated Chars	
-114	Timeout End Message	
-115	Timeout Connect	
-200	Protocol Error	• Check if the PLC model is compatible with driver documentation • Check the configured Address field in **Device Points**
-201	Invalid Protocol	• Check if the PLC model is compatible with driver documentation • Contact technical support
-202	Invalid Station	• Check configured IP Address field in **Device Node** • Restart the driver
-204	Invalid Message Sequence	• Check if the PLC model is compatible with driver documentation • Check the configured Address field in **Device Points**
> 0	CIP Error	• See CIP error codes table

CIP Error Codes

CIP Error Codes	
Error Code	**Description**
1	Connection Failure.
2	Insufficient resources.
3	Value invalid.
4	IOI could not be deciphered or tag does not exist.
5	Unknown destination.
6	Data requested would not fit in response packet.
7	Loss of connection.
8	Unsupported service.
9	Error in data segment or invalid attribute value.
10	Attribute list error.
11	State already exists.
12	Object model conflict.
13	Object already exists.
14	Attribute not settable.
15	Permission denied.
16	Device state conflict.
17	Reply will not fit.
18	Fragment primitive.
19	Insufficient command data / parameters specified to execute service.
20	Attribute not supported.
21	Too much data specified.
26	Bridge request too large.
27	Bridge response too large.
28	Attribute list shortage.
29	Invalid attribute list.
30	Embedded service error.
31	Failure during connection.
34	Invalid reply received.
37	Key segment error.
38	Number of IOI words specified does not match IOI word count.
39	Unexpected attribute in list.

Troubleshooting tips

In this driver, it is very important to enable the TraceWindow messages, as invalid addresses can cause all the communication block with the PLC to fail, the TraceWindow tool will display the first invalid address found on the block when Device is enabled on the settings.

In order to have a quick view on the many communication blocks, open the ModuleInformation, navigate on the tree to find ControlLogix and then select Read Groups. Looking at the number and success and fail communication counters, you can easily identify if there is a block with error and then use the TraceWindow to locate the wrong address.

Driver Revision History

ControlLogix Driver Revision History	
Version	**Notes**
1.9.1.9	Program-scope tags now communicate; new per-point CIPLittleEndian option lets the ByteSwap modifier control DINT/INT/REAL byte order; strings addressed through LEN and DATA member points are handled correctly; L5K import keeps two- and three-dimensional arrays, types Float/Decimal I/O tags as REAL/DINT, accepts paths containing ';', and reports an error for L5X or non-L5K files instead of importing nothing; online import gives COUNTER members in user types the correct type; whole-array points now include the last element of every dimension (check the Array size of hand-built array tags), and an array read the controller rejects marks every position bad; new ControlLogix Simulator on Devices > Channels; optional off-by-default corrections are available through support.
1.9.1.1	10.1.5 release. Added `CIPLittleEndian` as a per-Point address option (third field of the Point Address syntax: `<Type>:<DeviceTagName>:<CIPLittleEndian>`) so byte order can be controlled at the individual Point level rather than only at Channel scope. When `CIPLittleEndian` is enabled, the driver handles byte ordering automatically and any **ByteSwap** modifier on the Point has no effect; disable `CIPLittleEndian` on the Point to use ByteSwap modifiers. Byte-swap correction on CIP little-endian read path. Documented Program-scope tag syntax (`Program:<ProgramName>.<TagName>`).
1.9.0.3	Initial release on new documentation standards

In this section...