

# Omron FINS Driver

<b>File Name</b>	FINS.dll
<b>Manufacturer</b>	Omron Corporation
<b>Devices</b>	CS, CJ, and CV models
<b>Protocol</b>	FINS (Ethernet/UDP) and Host Link (Serial or Ethernet/TCP)
<b>Version</b>	2.0.8
<b>Last Update</b>	04/02/2026
<b>Platform</b>	Win32
<b>Dependencies</b>	IOKit version 2.0
<b>Superblock Readings</b>	No
<b>Level</b>	0

## Introduction

The goal of the Omron FINS Driver is to communicate with programmable logic controllers by Omron Corporation, which can be placed on multiple network layers in a server computer, either via **Ethernet** or **Serial** interface.

## Driver Settings

The FINS command system supports communication with devices on local and remote Omron networks.

Driver OMRON FINS v2.0.6 (IOKit v2.0.130)

FINS Setup Serial Ethernet Modem RAS

FINS addresses (PLC):

Network:	Node:	Unit:
0	0	0

Source addresses (PC):

Network:	Node:	Unit:
0	0	0

N1/B1 Network.Node.Unit

N1/B1	Network	Node	Unit

Add Update Remove

PLC Type: CS/CJ

Ignore non-fatal CPU unit errors

OK Cancel Apply

**FINS tab**

The properties window of this Driver, in addition to several other configuration options, also displays a selectable list with identifiers of remote devices already configured. This list appears empty if there are no configured identifiers.

To determine a list of identifiers from remote devices, users must fill in correctly all options in the **FINS addresses (PLC)** group.

A device identifier is a network address with 3 (three) layers, which uniquely identify a target device. The format of a device identifier is **AAA.NNN.UU**, in which:

- **AAA**: Address of a FINS destination network configured in the **Network** option, a value between 0 (zero) and 127
- **NNN**: Number of a FINS destination node configured in the **Node** option, a value between 0 (zero) and 255
- **UU**: Number of a Host Link unit used for the interface of a computer, configured in the **Unit** option, a value between 0 (zero) and 31

The next options are available for manipulating data on the list of identifiers:

- **Add**: Adds new parameters
- **Update**: Changes parameters already listed
- **Remove**: Completely removes a row of parameters

When adding a new identifier to this list, it automatically receives an index. This index is then used to refer to this identifier when configuring a Tag.

The **Source addresses (PC)** group refers to the configuration of the local computer and its options are configured the same way as the **FINS addresses (PLC)** group. The **PLC Type** option allows selecting the type of device to communicate. Possible values for this property are **CS/CJ** or **CV**.

The **Ignore non-fatal CPU unit errors** option can be selected if non-fatal error codes are returned by a device. It is normal for this Driver to consider every non-fatal error as an error, discarding the values received despite being indicated as data for a normal completion and not compromising the integrity of these values. When selecting this option, this Driver handles with less rigor and disregard the indication of non-fatal errors in a unit.

#### NOTE

Information to fill in the **Source addresses (PC)** group are **Network**: Must be the same as the local device, **Node**: Must be different from local device's node and different from 0 (zero), and **Unit**: Not used and it is recommended to leave it in 0 (zero). For communication via Ethernet with devices using the FINS protocol, the **Transport** option configured on the **Ethernet** tab must be equal to **UDP/IP**. For communication via Ethernet with devices using the Host Link protocol, the **Transport** option configured on the **Ethernet** tab must be equal to **TCP/IP**.

## FINS Networks

Basic communication concepts of FINS networks needed to configure this Driver are discussed on the next topics. Please check Omron Corporation's documentation for a more detailed discussion about FINS networks.

## FINS Purpose

The FINS communication service was developed by Omron Corporation so that devices and computers on different networks can communicate with each other in a consistent way. Among compatible network types there are **Ethernet**, **Host Link**, **Controller Link**, **SYSMAC LINK**, **SYSMAC WAY**, and **Toolbus**. The FINS communication service allows communication among nodes with up to 3 (three) depth layers. Direct links between a computer and a device through a Server link are not considered as network layers.

The diagram on the next figure shows a FINS network composed by interconnected **SYSMAC LINK** and **Controller Link** networks, and it serves as an example for the following discussion. The FINS communication service allows communication

between any pair of devices on the diagram. **PLC 1** device works as server's **Host Link** interface to network 1 (one). **PLC 2** device works as a gateway between networks 1 (one) and 2 (two). In this example, the server computer sends a data request command, shown in red, to **PLC 4** device via **PLC 1** and **PLC 2** devices. The response is shown in blue. Initially, **PLC 4** or **PLC 5** devices could work as gateways to a third network layer, which could also be reached by a server computer.

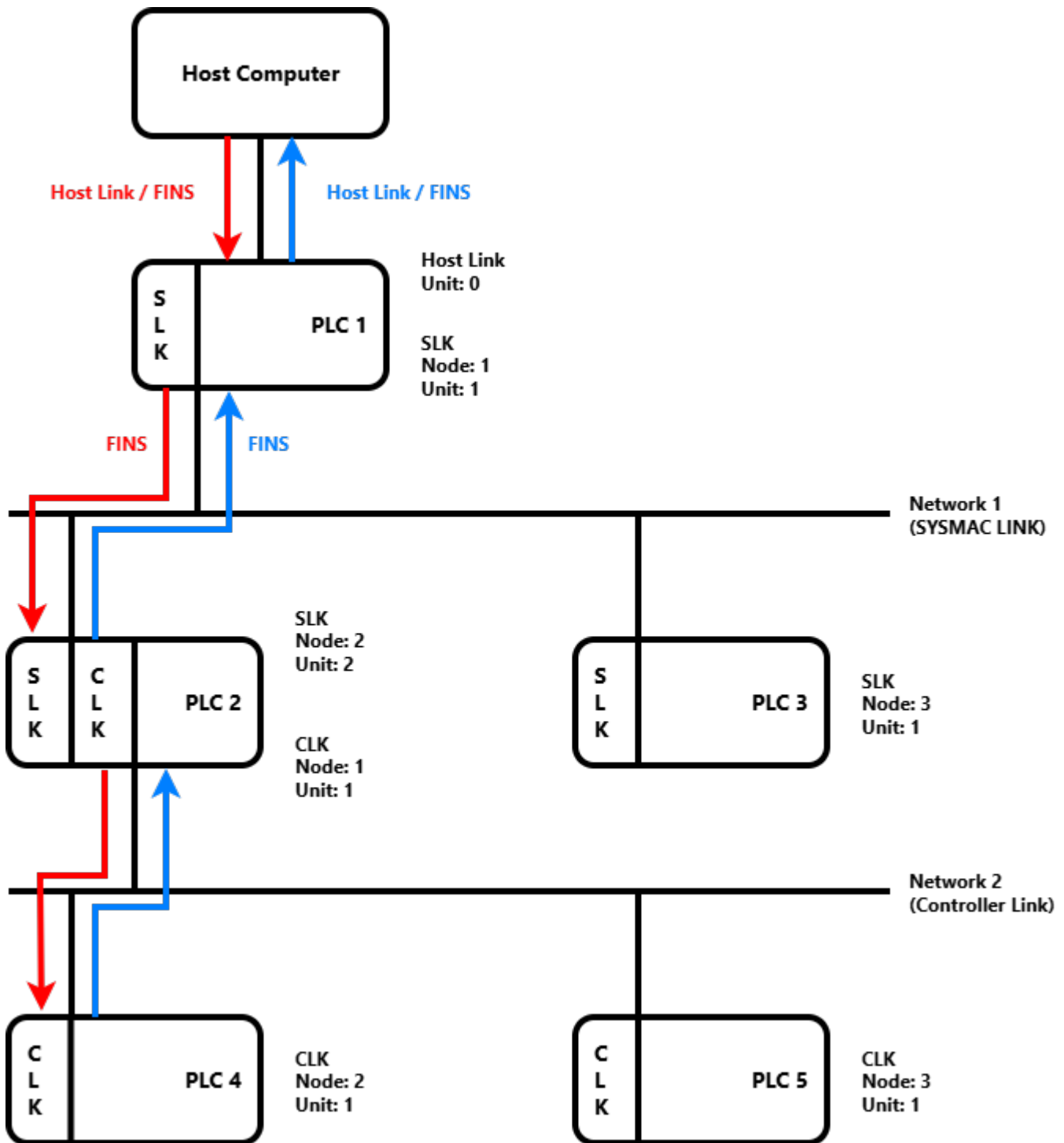


Diagram of a FINS network

## FINS Messages

FINS messages are composed by a header and a data portion. The header contains, among other information, data about source and destination. Data portion contains command codes and parameters for optional commands. Particularly interesting here are the source and destination parameters in the header, described next:

- **DNA:** Address of a destination network

- **DA1**: Number of a destination node
- **DA2**: Address of a destination module
- **SNA**: Address of a source network
- **SA1**: Number of a source node
- **SA2**: Address of a source module

This Driver always configures **DA2** or **SA2** parameters as 0 (zero). This means that communication must be performed between the host computer and the destination node of the **CPU** module. This Driver also configures **SNA** and **SA1** parameters as 0 (zero), indicating that communication must be performed through the **Host Link** port of the interface device. The interface device, in this case **PLC 1**, reconfigures **SNA** and **SA1** parameters as needed, when forwarding messages to other devices.

## Host Link Communication

This Driver can communicate with any FINS-compatible device through a **Host Link** port. That device can then process a given FINS command if it is the destination node, or relay the message to another device if it is not the destination node.

According to the **Diagram of a FINS network** figure on topic **FINS Purpose**, it is advisable to send a data request from the server computer to the **PLC 4** device. This Driver builds a FINS data request message with the appropriate source and destination parameters in its header, encapsulates that message in a **Host Link** package, and sends it to the **PLC 1** device. This device then examines the FINS message header and determines that node 2 (two) or network 2 (two), that is, the **PLC 4** device, is the intended destination.

When referring to its routing tables, discussed shortly, the **PLC 1** device determines that the message must be sent to the gateway device, **PLC 2**, to reach the destination node on network 2 (two). As the **PLC 2** device is not on a **Host Link** network, the **Host Link** package is removed. The **PLC 2** device, by its turn, sends the message to the **PLC 4** device. This device then sends its response back to the server computer via **PLC 1** and **PLC 2** devices. The **PLC 1** device wraps the FINS response in a **Host Link** package, which this Driver can recognize and process. Similar device objects could be created for devices **PLC 1**, **PLC 2**, **PLC 3**, and **PLC 5**.

## Routing Tables

With multi-level networks, every additional information must be programmed in the devices, so that they can send messages to other nodes in the system. This is accomplished by using FINS routing tables. FINS routing tables are **Local** and **Remote**. Local routing tables associate a network number with one of the communication modules (SIU, or *Special Input or Output Units*) installed in the device's rack. Remote routing tables give directions on how to reach one of the next network levels. The routing tables for the previous example are described next.

### Local PLC 1 device

NETWORK NUMBER	UNIT
1	1 (one)

As the **PLC 1** device is only connected to the **SYSMAC LINK** network, it has only one entry in its local routing table.

### Remote PLC 1 device

NETWORK NUMBER	RELAY NETWORK	RELAY NODE
2	1 (one)	2 (two)

Network 2 (two) is a remote network for the **PLC 1** device. To send a message to a node on network 2 (two), this device must send that message to a relay node, that is, the gateway on one of its local networks. This local network is called a *relay network*, and it must be network 1 (one), in this case. The **SYSMAC LINK** module in the gateway, the **PLC 2** device, is the node number 2 (two). Therefore, the relay node for the **PLC 1** device is equal to 2 (two).

#### Local PLC 2 device

NETWORK NUMBER	UNIT
1	2 (two)
2	1 (one)

There are two local networks for the **PLC 2** device. The **SYSMACK LINK** and **Controller Link** network numbers are arbitrarily assigned as unit numbers 2 (two) and 1 (one), respectively. The **SYSMAC LINK** and **Controller Link** modules are assigned as unit numbers 2 (two) and 1 (one), respectively. The network node number of each one of these modules is configured by the user, and it must be unique within its associated network. Both communication modules in the **PLC 2** device could be named as "node 10", for example, as they are on different networks.

#### Local PLC 3 device

NETWORK NUMBER	UNIT
1	1 (one)

The local routing table for the **PLC 3** device contains only one entry, as this device is only connected to network 1 (one).

#### Remote PLC 3 device

REMOTE NETWORK NUMBER	RELAY NETWORK	RELAY NODE
2	1 (one)	2 (two)

Network 2 (two) is a remote network for the **PLC 3** device. As in the **PLC 1** device, the relay network is 1 (one), and the relay node is 2 (two, the **SLK** module of the **PLC 2** device's gateway).

Routing tables for **PLC 4** and **PLC 5** devices must look like the following tables. In this example, they are the same because **Controller Link** modules in both devices are number 1 (one) units. Users must assign node numbers to these modules that are unique in network 2 (two).

#### Local PLC 4 and PLC 5 devices

NETWORK NUMBER	UNIT
2	1 (one)

#### Remote PLC 4 and PLC 5 devices

NETWORK NUMBER	RELAY NETWORK	RELAY NODE
1	2 (two)	1 (one)

# Tag Reference

This section contains information about the configuration of **[N/B]** parameters of this Driver.

## [N/B] Parameters for Reading

<b>N1 or B1</b>	Index for the address of a device, configured on the <b>Configuration Window of this Driver</b>
<b>N2 or B2</b>	Command in <b>TANN</b> format, in which <b>T</b> is a <b>data type</b> , <b>A</b> is an <b>area type</b> , and <b>NN</b> is a <b>command</b>
<b>N3 or B3</b>	Starting address for reading
<b>N4 or B4</b>	Starting bit for reading if data type is equal to <b>Bit</b> or the size of a <b>String</b> if data type is equal to <b>String</b> . Examples: <ul style="list-style-type: none"> <li>• <b>D0123, reading Words:</b> <math>N2 = 111, N3 = 123</math></li> <li>• <b>A10.15, Bit:</b> <math>N2 = 103, N2 = 10, N4 = 15</math></li> <li>• <b>D10, String, size 20:</b> <math>N2 = 7111, N3 = 10, N4=20</math></li> </ul>

## Data Types

### Data types (thousands)

VALUE	DATA TYPE
0	<b>Word</b> or <b>Bit</b> , depending on the type of <b>command</b>
1	<b>Short</b>
2	<b>BCD</b>
3	<b>Long</b>
4	<b>DWord</b>
5	<b>LBCD</b>
6	<b>Float</b>
7	<b>String</b> with high-low byte order
8	<b>String</b> with low-high byte order

## Area Types

### Area types (hundreds)

VALUE	AREA TYPE
1NN	Reading of a memory area
200	Reading of a program area

## Available Commands

### Available commands (tenths and units)

VALUE	COMMAND
100	CIO BIT (CIO)

VALUE	COMMAND
101	WORK BIT (WK)
102	HOLD BIT (HR)
103	AUX BIT (AR)
104	CIO WORD (CIO)
105	WORK WORD (WK)
106	HOLD WORD (HR)
107	AUX WORD (AR)
108	TIMERCOUNTER FLAG
109	TIMERCOUNTER PV
110	DM BIT
111	DM WORD
112	TASK BIT (TK)
113	TASK STATUS (TK)
114	IDXREG PV (IR)
115	DATREG PV (DR)
116	CLKCOND BIT

#### Commands exclusive to CS and CJ models

VALUE	COMMAND
117	EM BANK 0 ( <b>Word</b> )
118	EM BANK 1 ( <b>Word</b> )
119	EM BANK 2 ( <b>Word</b> )
120	EM BANK 3 ( <b>Word</b> )
121	EM BANK 4 ( <b>Word</b> )
122	EM BANK 5 ( <b>Word</b> )
123	EM BANK 6 ( <b>Word</b> )
124	EM BANK 7 ( <b>Word</b> )
125	EM BANK 8 ( <b>Word</b> )
126	EM BANK 9 ( <b>Word</b> )
127	EM BANK A ( <b>Word</b> )
128	EM BANK B ( <b>Word</b> )
129	EM BANK C ( <b>Word</b> )
130	EM CURRENT BANK
131	EM CURRENT BANK NUMBER
132	EM BANK 0 ( <b>Bit</b> )
133	EM BANK 1 ( <b>Bit</b> )

VALUE	COMMAND
134	EM BANK 2 ( <b>Bit</b> )
135	EM BANK 3 ( <b>Bit</b> )
136	EM BANK 4 ( <b>Bit</b> )
137	EM BANK 5 ( <b>Bit</b> )
138	EM BANK 6 ( <b>Bit</b> )
139	EM BANK 7 ( <b>Bit</b> )
140	EM BANK 8 ( <b>Bit</b> )
141	EM BANK 9 ( <b>Bit</b> )
142	EM BANK A ( <b>Bit</b> )
143	EM BANK B ( <b>Bit</b> )
144	EM BANK C ( <b>Bit</b> )

#### Commands exclusive to CV models

VALUE	COMMAND
117	EM BANK 0 ( <b>Word</b> )
118	EM BANK 1 ( <b>Word</b> )
119	EM BANK 2 ( <b>Word</b> )
120	EM BANK 3 ( <b>Word</b> )
121	EM BANK 4 ( <b>Word</b> )
122	EM BANK 5 ( <b>Word</b> )
123	EM BANK 6 ( <b>Word</b> )
124	EM BANK 7 ( <b>Word</b> )
125	EM CURRENT BANK
126	EM CURRENT BANK NUMBER

## [N/B] Parameters for Writing

N1 or B1	Index for the address of a device, configured on the <b>Configuration Window of this Driver</b>
N2 or B2	Command in <b>TANN</b> format, in which <b>T</b> is a <b>data type</b> , <b>A</b> is a <b>command type</b> , and <b>NN</b> is a <b>command</b>
N3 or B3	Starting address for writing
N4 or B4	Starting bit for writing if data type is equal to <b>Bit</b> or the size of a <b>String</b> if data type is equal to <b>String</b>

## Data Types

#### Data types (thousands)

VALUE	DESCRIPTION
0	<b>Word</b> or <b>Bit</b> , depending on the type of <b>command</b>
1	<b>Short</b>
2	<b>BCD</b>
3	<b>Long</b>
4	<b>DWord</b>
5	<b>LBCD</b>
6	<b>Float</b>
7	<b>String</b> with high-low byte order
8	<b>String</b> with low-high byte order

## Command Types

### Command types (hundreds)

VALUE	DESCRIPTION
1NN	Writing of a memory area
200	Writing of a program area
3NN	Forced Set or Reset. For more information, please check topic <b>Forced Set or Reset Commands</b>
4NN	Other commands. For more information, please check topic <b>Other Commands</b>

## Writing Commands

### Writing commands (tenths and units)

VALUE	COMMAND
100	CIO BIT (CIO)
101	WORK BIT (WK)
102	HOLD BIT (HR)
103	AUX BIT (AR)
104	CIO WORD (CIO)
105	WORK WORD (WK)
106	HOLD WORD (HR)
107	AUX WORD (AR)
108	TIMERCOUNTER FLAG
109	TIMERCOUNTER PV
110	DM BIT
111	DM WORD
112	TASK BIT (TK)
113	TASK STATUS (TK)

VALUE	COMMAND
114	IDXREG PV (IR)
115	DATREG PV (DR)
116	CLKCOND BIT

**Commands exclusive to CS and CJ models**

VALUE	COMMAND
117	EM BANK 0 ( <b>Word</b> )
118	EM BANK 1 ( <b>Word</b> )
119	EM BANK 2 ( <b>Word</b> )
120	EM BANK 3 ( <b>Word</b> )
121	EM BANK 4 ( <b>Word</b> )
122	EM BANK 5 ( <b>Word</b> )
123	EM BANK 6 ( <b>Word</b> )
124	EM BANK 7 ( <b>Word</b> )
125	EM BANK 8 ( <b>Word</b> )
126	EM BANK 9 ( <b>Word</b> )
127	EM BANK A ( <b>Word</b> )
128	EM BANK B ( <b>Word</b> )
129	EM BANK C ( <b>Word</b> )
130	EM CURRENT BANK
131	EM CURRENT BANK NUMBER
132	EM BANK 0 ( <b>Bit</b> )
133	EM BANK 1 ( <b>Bit</b> )
134	EM BANK 2 ( <b>Bit</b> )
135	EM BANK 3 ( <b>Bit</b> )
136	EM BANK 4 ( <b>Bit</b> )
137	EM BANK 5 ( <b>Bit</b> )
138	EM BANK 6 ( <b>Bit</b> )
139	EM BANK 7 ( <b>Bit</b> )
140	EM BANK 8 ( <b>Bit</b> )
141	EM BANK 9 ( <b>Bit</b> )
142	EM BANK A ( <b>Bit</b> )
143	EM BANK B ( <b>Bit</b> )
144	EM BANK C ( <b>Bit</b> )

### Commands exclusive to CV models

VALUE	COMMAND
117	EM BANK 0 ( <b>Word</b> )
118	EM BANK 1 ( <b>Word</b> )
119	EM BANK 2 ( <b>Word</b> )
120	EM BANK 3 ( <b>Word</b> )
121	EM BANK 4 ( <b>Word</b> )
122	EM BANK 5 ( <b>Word</b> )
123	EM BANK 6 ( <b>Word</b> )
124	EM BANK 7 ( <b>Word</b> )
125	EM CURRENT BANK
126	EM CURRENT BANK NUMBER

## Forced Set or Reset Commands

### Forced Set or Reset commands (tenths and units)

VALUE	COMMAND
300	FORCED RESET
301	FORCED SET
302	FORCED OFFRELEASE
303	FORCED ONRELEASE
304	FORCED RELEASE

## Other Commands

### Other commands (tenths and units)

VALUE	COMMAND
400	<b>Run</b> mode
401	<b>Monitor</b> mode
402	<b>Stop</b> mode
403	Cancels a forced Set or Reset command

## Documentation of I/O Interfaces

This section contains the documentation of I/O Interfaces referring to the **FINS** Driver.

## Configuration of a Driver

I/O Interface configuration is performed on a Driver's configuration dialog box. To access the configuration of this dialog box in **Eclipse E3** in version 1.0, follow these steps:

1. Right-click a Driver object (IODriver).
2. Select the **Properties** item on the contextual menu.
3. Select the **Driver** tab.
4. Click **Other parameters**.

In **Eclipse E3** version 2.0 or later, click **Configure driver**  on a Driver's toolbar. In **Eclipse SCADA**, follow these steps:

1. Open the Organizer.
2. Select a Driver on Organizer's tree.
3. Click **Extras** on the **Driver** tab.

Currently, an I/O Interface allows opening only one connection for each Driver. This means that, if users want to access two serial ports, they must add two Drivers to an application and then configure each one of these Drivers for each serial port.

## Configuration Dialog Box

The dialog box of I/O Interfaces allows configuring the I/O connection used by a Driver. This dialog box contains the **Setup**, **Serial**, **Ethernet**, **Modem**, and **RAS** tabs, described on the next topics. If a Driver does not implement a specific I/O connection, its corresponding tab is not available for configuration. Some Drivers may contain additional tabs, specific for that Driver, on the configuration dialog box.

## Setup Tab

The **Setup** tab contains general configurations of a Driver. This tab is divided into the following groups:

- **General configurations:** Configurations of a Driver's physical layer, time-out, and initialization mode
- **Connection management:** Configurations on how the I/O Interface keeps a connection and which recovery policy is used on failure
- **Logging options:** Controls the generation of log files

Setup

Physical Layer: Ethernet  Start driver OFFLINE

Timeout: 1000 ms    Communication check time: 5000 ms

Connection management

Mode: Automatic (managed by the driver)

Retry failed connection every 20 seconds

Give up after 1 failed retries

Disconnect if non-responsive for 0 seconds

Logging Options

Log to File: C:\eeLogs\MicrolokII\_%DATE%.log

File size limit (MB): 0 ('0' is unlimited)

**Setup tab**

**General options on the Setup tab**

OPTION	DESCRIPTION
<b>Physical Layer</b>	Select the physical layer on a list. Available options are <b>Serial</b> , <b>Ethernet</b> , <b>Modem</b> , and <b>RAS</b> . The selected interface must be configured on its specific tab
<b>Timeout</b>	Configure a time-out, in milliseconds, for the physical layer. This is the amount of time an I/O interface waits to receive any byte from the reception's buffer
<b>Communication check time</b>	Set the time, in milliseconds, to define the interval at which communication is considered to be in an inactive state. As long as an I/O Driver receives valid data, its communication state is considered active. However, if during operation an I/O Driver does not receive valid data inside this period of time, the state is considered inactive. The communication state is shown in the <b>IO.CommunicationStatus</b> Tag
<b>Start driver OFFLINE</b>	Select this option so that a Driver starts in <b>Offline</b> mode or stopped. This means that the I/O interface is not created until this Driver is configured to <b>Online</b> mode by using a Tag in an application. This mode enables a dynamic configuration of an I/O interface at run time

## Options on the Connection management group

OPTION	DESCRIPTION
<b>Mode</b>	Selects a management mode of a connection. Selecting the <b>Automatic</b> option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the <b>Manual</b> option allows an application to fully manage a connection
<b>Retry failed connection every ... seconds</b>	Select this option to enable a Driver's connection retry in a certain interval, in seconds. If the <b>Give up after failed retries</b> option is not selected, this Driver keeps retrying until a connection is performed, or until the application is stopped
<b>Give up after ... failed retries</b>	Enable this option to define a maximum number of connection retries. When the specified number of consecutive connection retries is reached, a Driver goes to the <b>Offline</b> mode, assuming that a hardware problem was detected. If a Driver establishes a successful connection, the number of unsuccessful retries is cleared. If this new connection is lost, then the retry counter starts at zero
<b>Disconnect if non-responsive for ... seconds</b>	Enable this option to force a Driver to disconnect if no byte was received by the I/O interface during the specified time-out, in seconds. This time-out must be greater than the time-out configured in the <b>Timeout</b> option

Options on the Logging Options group

OPTION	DESCRIPTION
<p><b>Log to File</b></p>	<p>Enable this option and configure the name of a file to write a log. Log files can be large, so use this option for short periods of time, only for testing and debugging purposes. If the <b>%PROCESS%</b> macro is used in the log file name, it is replaced by the identifier of the current process. This option is particularly useful when using several instances of the same Driver in <b>Elipse E3</b>, thus allowing each instance to generate a separate log file. For example, when configuring this option with value "c:\e3logs\drivers\sim_%PROCESS%.log", it generates a file named c:\e3logs\drivers\sim_00000FDA.log for process <b>OFDAh</b>. Users can also use the <b>%DATE%</b> macro in the file name. In this case a log file is generated every day, in the format <b>aaaa_mm_dd</b>. For example, when configuring this option with value "c:\e3logs\drivers\sim_%DATE%.log", it generates a file named c:\e3logs\drivers\sim_2005_12_31.log in 12/31/2005 and a file named c:\e3logs\drivers\sim_2006_01_01.log in 01/01/2006. Similarly, the <b>%DATE_HOUR%</b> macro generates one log file per hour, in the format <b>aaaa_mm_dd_hh</b></p>
<p><b>File size limit (MB)</b></p>	<p>Configure the log file size limit, in megabytes. A value equal to 0 (zero) means that there is no size limit for the log file</p>

## Serial Tab

Use this tab to configure parameters for a **Serial** Interface.

Serial

Port:

Baud rate:

Data bits:

Parity:

Stop bits:

Enable 'ECHO' suppression

Handshaking

DTR control:

RTS control:

Wait for CTS before send

CTS timeout:  ms

Delay before send:  ms

Delay after send:  ms

Inter-byte delay (microseconds):   $\mu$ s

Inter-frame delay (milliseconds):  ms

Serial tab

General options on the Serial tab

OPTION	DESCRIPTION
<b>Port</b>	Select a serial port on the list, from <b>COM1</b> to <b>COM4</b> , or type the name of a serial port in the format <b>COMn</b> , such as "COM15". When typing the name of a serial port manually, the dialog box only accepts names of serial ports starting with the expression "COM"
<b>Baud rate</b>	Select a baud rate on the list ( <b>1200, 2400, 4800, 9600, 19200, 38400, 57600, or 115200</b> ) or type a baud rate, such as 600
<b>Data bits</b>	Select 7 (seven) or 8 (eight) data bits on the list
<b>Parity</b>	Select a parity on the list. The available options are <b>None, Even, Odd, Mark, or List</b>
<b>Stop bits</b>	Select the number of stop bits on the list. The available options are <b>1, 1.5, or 2</b> stop bits
<b>Enable 'ECHO' suppression</b>	Enable this option to remove the echo received after the I/O Interface sends data via serial port. If this echo is not equal to the bytes just sent, then the I/O Interface aborts communication
<b>Inter-byte delay (microseconds)</b>	Defines a delay between each byte transmitted by the I/O Interface, in millionths of a second, that is, 1000000 is equal to a second. This option must be used with small delays of less than a millisecond
<b>Inter-frame delay (milliseconds)</b>	Defines a delay between packets sent or received by the I/O Interface, in thousandths of a second, that is, 1000 is equal to a second. This delay is applied if the I/O Interface

OPTION	DESCRIPTION
	sends two consecutive packets, or between a received packet and the next sending

The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process, that is, it controls when data can be sent or received via serial line. Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works with **RS232**-type serial lines as well as with **RS485**-type serial lines.

#### Available options on the Handshaking group

OPTION	DESCRIPTION
<b>DTR control</b>	Select the value <b>ON</b> to keep the <b>DTR</b> signal always on while the serial port is open. Select the value <b>OFF</b> to turn the <b>DTR</b> signal off while the serial port is open. Some devices require the <b>DTR</b> signal always on to allow communication
<b>RTS control</b>	Select the value <b>ON</b> to keep the <b>RTS</b> signal always on while the serial port is open. Select the value <b>OFF</b> to turn the <b>RTS</b> signal off while the serial port is open. Select the value <b>Toggle</b> to turn the <b>RTS</b> signal on while sending bytes via serial port and turn it off when not sending bytes, therefore enabling the reception
<b>Wait for CTS before send</b>	Available only when the <b>RTS control</b> option is configured with the value <b>Toggle</b> . Use this option to force a Driver to check the <b>CTS</b> signal before sending bytes via serial port, after turning the <b>RTS</b> signal on. In this mode, the <b>CTS</b> signal is handled as a permission flag for sending
<b>CTS timeout</b>	Determines a maximum time, in milliseconds, that a Driver waits for the <b>CTS</b> signal after turning the <b>RTS</b> signal on. If the <b>CTS</b> signal is not turned on within this time-out, that Driver then fails the current communication and returns an error
<b>Delay before send</b>	Some serial port devices have a delay when enabling a data sending circuit after the <b>RTS</b> signal is turned on. Configure this option to wait a certain number of milliseconds after turning the <b>RTS</b> signal on and before sending the first byte. <b>IMPORTANT</b> : This delay must be used carefully, because it uses 100% of CPU resources while waiting. System's general performance degrades as this value increases
<b>Delay after send</b>	This is the same effect of the <b>Delay before send</b> option, but in this case the delay is performed after sending the last byte, before turning the <b>RTS</b> signal off

## Ethernet Tab

Use this tab to configure parameters of an **Ethernet** Interface. These parameters, except port configurations, must also be configured for use in the **RAS** Interface.

Ethernet

Transport: TCP/IP ▼

PING before connecting

Timeout: 4000 ms

Retries: 1

Listen for connections on port: 0

Share listen port with other processes

Interface: (All Interfaces) ▼

Use IPv6  Use SSL SSL Settings

Enable 'ECHO' supression

IP Filter:

Connect to

<input type="checkbox"/> Main IP:	<span style="border: 1px solid gray; padding: 2px;"> </span>	Port:	<span style="border: 1px solid gray; padding: 2px;">502</span>	<input type="checkbox"/> Local port:	<span style="border: 1px solid gray; padding: 2px;">0</span>
<input type="checkbox"/> Backup IP 1:	<span style="border: 1px solid gray; padding: 2px;"> </span>	Port:	<span style="border: 1px solid gray; padding: 2px;">0</span>	<input type="checkbox"/> Local port:	<span style="border: 1px solid gray; padding: 2px;">0</span>
<input type="checkbox"/> Backup IP 2:	<span style="border: 1px solid gray; padding: 2px;"> </span>	Port:	<span style="border: 1px solid gray; padding: 2px;">0</span>	<input type="checkbox"/> Local port:	<span style="border: 1px solid gray; padding: 2px;">0</span>
<input type="checkbox"/> Backup IP 3:	<span style="border: 1px solid gray; padding: 2px;"> </span>	Port:	<span style="border: 1px solid gray; padding: 2px;">0</span>	<input type="checkbox"/> Local port:	<span style="border: 1px solid gray; padding: 2px;">0</span>

**Ethernet tab**

**Available options on the Ethernet tab**

OPTION	DESCRIPTION
<b>Transport</b>	Select the value <b>TCP/IP</b> for a TCP socket ( <i>stream</i> ) or select the value <b>UDP/IP</b> to use a UDP socket ( <i>connectionless datagram</i> )
<b>Listen for connections on port</b>	Use this option to wait for new connections in a specific IP port, common in Slave Drivers. If this option remains unselected, a Driver connects to the address and port specified in the <b>Connect to</b> option
<b>Share listen port with other processes</b>	Select this option to share the listening port with other Drivers and processes
<b>Interface</b>	Select the local network interface, identified by its IP address, that a Driver uses to establish and receive connections, or select the value <b>(All Interfaces)</b> to allow connection in any network interface
<b>Use IPv6</b>	Select this option to force a Driver to use addresses in <b>IPv6</b> format on all Ethernet connections. Leave this option deselected to use the <b>IPv4</b> format
<b>Enable 'ECHO' supression</b>	Enable this option to remove the echo from received data. An echo is a copy of sent data, which can be returned before a reply message
<b>IP Filter</b>	List of restricted or allowed IP addresses from where a Driver accepts connections ( <i>Firewall</i> ). Please check the <b>IO.Ethernet.IPFilter</b> property for more information
<b>PING before connecting</b>	Enable this option to execute a <b>ping</b> command, that is, to check whether a device can be reached on a network, for a device before trying a socket connection. This is a quick way

OPTION	DESCRIPTION
	<p>of determining a successful connection before trying to open a socket with a device. The time-out of a connection with a socket can be very high. The available options are:</p> <ul style="list-style-type: none"> <li>• <b>Timeout:</b> Specify the number of milliseconds to wait for a reply from a <b>ping</b> command. Users must use a <b>ping</b> command to check the normal reply time, configuring this option for a value above that average. Usually this value can be configured between 1000 and 4000 milliseconds, that is, between 1 (one) and 4 (four) seconds</li> <li>• <b>Retries:</b> Number of retries of a <b>ping</b> command, not counting the first attempt. If all attempts fail, then the socket connection is aborted</li> </ul>

**Available options on the Connect to group**

OPTION	DESCRIPTION
<b>Main IP</b>	Type the IP address of a remote device. Users can use an IP address separated by dots, as well as a URL. In case of a URL, a Driver uses the available DNS service to map that URL to an IP address, such as "192.168.0.13" or "Server1"
<b>Port</b>	Type the IP port of a remote device, between 0 (zero) and 65535
<b>Local port</b>	Select this option to use a fixed local IP port when connecting to a remote device
<b>Backup IP 1, 2, and 3</b>	Indicate the IP address, the IP port, and the fixed local IP port of up to 3 (three) backup addresses of a remote device

## Modem Tab

Use this tab to configure parameters of a **Modem** Interface. Some options on the **Serial** tab affect the configuration of a modem, therefore users must also configure the **Serial** Interface.

Modem

Select the modem to use:

Dial Number:

Accept incoming calls

**Modem tab**

The **Modem** Interface uses the TAPI modems installed on the computer.

#### Available options on the Modem tab

OPTION	DESCRIPTION
<b>Select the modem to use</b>	Select a modem on the list of available modems on the computer. If the value <b>Default modem</b> is selected, then the first available modem is used. Selecting this option is recommended specially when an application is used on another computer
<b>Modem settings</b>	Click to open the configuration window of the selected modem
<b>Dial Number</b>	Type a default number for dialing. This value can be changed at run time. Users can use the <b>w</b> character to represent a pause or a waiting time for a dial tone. For example, "0w33313456" dials the number 0 (zero), waits, and then dials the number "33313456"
<b>Accept incoming calls</b>	Enable this option so that a Driver answers the phone when receiving an external call. To use this option, users must configure the <b>Connection management</b> option on the <b>Setup</b> tab to the value <b>Manual</b>

## RAS Tab

Use this tab configure parameters of a **RAS** Interface. Users must also configure the **Ethernet** tab.

A **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

When connecting to a RAS device, first the I/O Interface **IOKit** connects to the socket on the IP address and port configured on the **Ethernet** tab. After opening the socket, the following initialization or connection steps are performed:

1. Clears the socket, that is, removes any **TELNET** greeting message received from a RAS device.
2. Sends an **AT** dial message, in **ASCII** format, in the socket.
3. Waits for a **CONNECT** reply.
4. If the time-out expires, the connection is aborted.
5. If the **CONNECT** reply is received within the time-out, the socket is available for communication with a device, that is, the connection was established.

If step 5 (five) is successful, then the socket behaves as a normal socket, with the RAS device working as a router between a Driver and the device. Bytes sent by a Driver are received by the RAS device and sent to the destination device using a modem. Bytes received by the modem's RAS device are sent back to a Driver using the same socket.

After establishing a connection, the **RAS** interface monitors data received by a Driver. If a "NO CARRIER" **String** is found, the socket is closed. If the RAS device does not send a **NO CARRIER** signal, the **RAS** Interface cannot detect when the modem connection between the RAS device and the final I/O device fails. To recover from this failure, users are strongly advised to enable the **Disconnect if non-responsive** option on the **Setup** tab.

RAS

AT command:

Connection timeout:  seconds

Other socket settings should be configured in the "Ethernet" tab!

**RAS tab**

**Available options on RAS tab**

OPTION	DESCRIPTION
<b>AT command</b>	A <b>String</b> with the full <b>AT</b> command used to dial to a destination device. For example, "ATDT33313456" dials by tone to number "33313456"
<b>Connection timeout</b>	Number of seconds to wait for a modem's <b>CONNECT</b> reply, after sending an <b>AT</b> command

## General Configurations

This section contains information about the configuration of general **I/O Tags** and **Properties** of I/O Interfaces.

### I/O Tags

#### General I/O Interfaces Tags (N2/B2 = 0)

The Tags described next are provided for all supported I/O Interfaces.

#### IO.CommunicationStatus

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Reading
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	0 (zero)
<b>N4 Parameter</b>	6 (six)
<b>String Configuration</b>	IO.CommunicationStatus

This Tag informs the communication status of a Driver. It indicates how communication works relative to receiving valid data within a time period arbitrated in the configuration. For more information, please check topic **Setup Tab**. Possible values are **0 - Inactive communication**: The Driver did not receive valid data or stopped receiving data after *n* milliseconds, as configured in the properties window, or **1 - Active communication**: The Driver is receiving valid data.

#### IO.IOKitEvent

<b>Type of Tag</b>	Block Tag
<b>Type of Access</b>	Read-Only
<b>B1 Parameter</b>	-1 (minus one)
<b>B2 Parameter</b>	0 (zero)
<b>B3 Parameter</b>	0 (zero)
<b>B4 Parameter</b>	1 (one)
<b>Size Property</b>	4 (four)
<b>ParamItem Property</b>	IO.IOKitEvent

This Block returns Driver events generated by several sources in I/O Interfaces. The **TimeStamp** property of this Block represents the moment this event occurred. The Block Elements are the following:

- **Element 0**: Type of event. Possible values are **0**: Information, **1**: Warning, or **2**: Error
- **Element 1**: Source of an event. Possible values are **0**: Driver (specific of a Driver), **-1**: IOKit (generic events of I/O Interfaces), **-2**: **Serial** Interface, **-3**: **Modem** Interface, **-4**: **Ethernet** Interface, or **-5**: **RAS** Interface
- **Element 2**: Error number, specific for each source of event

- **Element 3:** Message of an event, a **String** specific for each event

**NOTE**

A Driver keeps a maximum number of 100 events internally. If additional events are reported, older events are discarded.

**IO.PhysicalLayerStatus**

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	0 (zero)
<b>N4 Parameter</b>	2 (two)
<b>String Configuration</b>	IO.PhysicalLayerStatus

This Tag indicates the status of a physical layer. Possible values are the following:

- **0:** Physical layer stopped, that is, a Driver is in **Offline** mode, the physical layer failed when initializing, or exceeded the maximum number of reconnection attempts
- **1:** Physical layer started but not connected, that is, a Driver is in **Online** mode but the physical layer is not connected. If the **Connection management** option is configured with the value **Automatic**, the physical layer can be connecting, disconnecting, or waiting for a reconnection attempt. If the **Connection management** option is configured with the value **Manual**, then the physical layer remains in this status until forced to connect
- **2:** Physical layer connected, that is, the physical layer is ready for use. This **DOES NOT** mean a device is connected, only that the access layer is working

**IO.SetConfigurationParameters**

<b>Type of Tag</b>	Block Tag
<b>Type of Access</b>	Read-Only
<b>B1 Parameter</b>	-1 (minus one)
<b>B2 Parameter</b>	0 (zero)
<b>B3 Parameter</b>	0 (zero)
<b>B4 Parameter</b>	3 (three)
<b>Size Property</b>	2 (two)
<b>ParamItem Property</b>	IO.SetConfigurationParameters

Use this Tag to change any property of a Driver's configuration dialog box at run time.

This Tag works only while a Driver is in **Offline** mode. To start a Driver in **Offline** mode, select the **Start driver OFFLINE** option on that Driver's configuration dialog box. Users can write to a PLC Tag or to a Block Tag containing the parameters to change. Writing individual Block Elements is not supported, the whole Block must be written at once.

In **Elipse SCADA**, users must use a Block Tag. Every parameter to configure uses two Block Elements. For example, if users want to configure 3 (three) parameters, then the size of the Block must be 6 (six, 3 × 2). The first Element is the property's name, as a **String**, and the second Element is the property's value, according to the next example.

```
// 'Block' must be a Block Tag with automatic reading,
// scan reading, and automatic writings disabled.
// Configure all parameters
Block.element001 = "IO.Type" // Parameter 1
Block.element002 = "Serial"
Block.element003 = "IO.Serial.Port" // Parameter 2
Block.element004 = 1
Block.element005 = "IO.Serial.BaudRate" // Parameter 3
Block.element006 = 19200
// Writes the whole Block
Block.Write()
```

When using **Elipse E3**, the ability to create arrays at run time allows using an I/O Tag as well as a Block Tag. Users can use the **Write** method of a Driver to send the parameters directly to that Driver, without creating a Tag, according to the next example.

```
Dim arr(6)
' Configure all array elements
arr(1) = "IO.Type"
arr(2) = "Serial"
arr(3) = "IO.Serial.Port"
arr(4) = 1
arr(5) = "IO.Serial.BaudRate"
arr(6) = 19200
' There are two methods to send parameters
' Method 1: Using an I/O Tag
tag.WriteEx arr
' Method 2: Without using a Tag
Driver.Write -1, 0, 0, 3, arr
```

A variation of the previous example uses a bidimensional array.

```
Dim arr(10)
' Configure all array elements. Notice the array was resized
' to 10 elements. Empty array elements are ignored by a Driver
arr(1) = Array("IO.Type", "Serial")
arr(2) = Array("IO.Serial.Port", 1)
arr(3) = Array("IO.Serial.BaudRate", 19200)
Driver.Write -1, 0, 0, 3, arr
```

A Driver does not validate parameter names or passed values, therefore be careful when writing parameters and values. The **Write** method fails if the configuration array is incorrectly created. Users can check the log of a Driver or use the *writeStatus* parameter of the **WriteEx** method to find out the exact cause of an error.

```
Dim arr(10), strError
arr(1) = Array("IO.Type", "Serial")
arr(2) = Array("IO.Serial.Port", 1)
arr(3) = Array("IO.Serial.BaudRate", 19200)
If Not Driver.WriteEx -1, 0, 0, 3, arr, , , strError Then
    MsgBox "Failed configuring Driver parameters: " + strError
End If
```

## IO.WorkOnline

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Reading or Writing
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	0 (zero)
<b>N4 Parameter</b>	4 (four)
<b>String Configuration</b>	IO.WorkOnline

This Tag informs the current status of a Driver and allows starting or stopping the physical layer. Possible values are the following:

- **0 - Driver Offline:** Physical layer is closed or stopped. This mode allows a dynamic configuration of a Driver's parameters using the **IO.SetConfigurationParameters** Tag
- **1 - Driver Online:** Physical layer is open or executing. While in **Online** mode, the physical layer can be connected or disconnected and its current status can be checked using the **IO.PhysicalLayerStatus** Tag

In the next example, using **Eclipse E3**, a Driver is configured to **Offline** mode, its COM port is changed, and then configured to **Online** mode again.

```
'Configure to Offline mode
Driver.Write -1, 0, 0, 4, 0
'Change port to COM2
Driver.Write -1, 0, 0, 3, Array("IO.Serial.Port", 2)
'Configure to Online mode
Driver.Write -1, 0, 0, 4, 1
```

The **Write** method may fail when configuring a Driver to **Online** mode, that is, writing the value 1 (one). In this case, this Driver remains in **Offline** mode. The cause of failure can be:

- Type of physical layer incorrectly configured, probably an invalid value was configured in the **IO.Type** property
- This Driver may have run out of memory
- Physical layer probably did not create its working thread. Search the log file for a message "Failed to create physical layer thread!"
- Physical layer could not start. The cause of this failure depends on the type of physical layer. It can be an invalid serial port number, a failure when starting Windows Sockets, or a failure when starting TAPI (modem), among others. This cause is recorded on the log file

### IMPORTANT

Even if the configuration of a Driver to **Online** mode is successful, this does not necessarily mean the physical layer is ready to use, that is, ready to execute input and output operations with an external device. The **IO.PhysicalLayerStatus** Tag must be checked to ensure the physical layer is connected and ready for communication.

## Properties

These are general properties of all supported I/O Interfaces.

## IO.ConnectionMode

9 Controls the management mode of a Connection. Possible values are **0**: Automatic mode, in which a Driver manages the connection or **1**: Manual mode, in which an application manages the connection.

## IO.GiveUpEnable

☑ When configured to True, defines a maximum number of reconnection attempts. If all reconnection attempts fail, a Driver enters the **Offline** mode. When configured to False, a Driver tries until a reconnection is successful.

## IO.GiveUpTries

9 Number of reconnection attempts before this one is aborted. For example, if the value of this property is equal to 1 (one), a Driver tries only one reconnection when the connection is lost. If this one fails, this Driver enters the **Offline** mode.

## IO.InactivityEnable

☑ Configure to True to enable and to False to disable inactivity detection. The physical layer is disconnected if inactive for a certain period of time. The physical layer is considered inactive only if it is capable of sending data but not capable of receiving it back.

## IO.InactivityPeriodSec

9 Number of seconds to check for inactivity. If the physical layer is inactive for this period of time, it is then disconnected.

## IO.RecoverEnable

☑ Configure to True to enable a Driver to recover lost connections and to False to leave a Driver in **Offline** mode when a connection is lost.

## IO.RecoverPeriodSec

9 Delay time between two connection attempts, in seconds.

### NOTE

The first reconnection is executed immediately after a connection is lost.

## IO.StartOffline

☑ Configure to True to start a Driver in **Offline** mode and to False to start a Driver in **Online** mode.


### NOTE

It is pointless to change this property at run time, as it can only be changed when a Driver is already in **Offline** mode. To configure a Driver in **Online** mode at run time, write the value 1 (one) to the **IO.WorkOnline** Tag.

## IO.TimeoutMs

9 Defines a time-out for the physical layer, in milliseconds. One second is equal to 1000 milliseconds.

## IO.Type

 Defines the type of physical interface used by a Driver. Possible values are the following:

- **N or None:** Does not use a physical interface, that is, a Driver must provide a customized interface
- **S or Serial:** Uses a local serial port (COM $n$ )
- **M or Modem:** Uses a local modem, internal or external, accessed via TAPI (*Telephony Application Programming Interface*)
- **E or Ethernet:** Uses a TCP/IP or UDP/IP socket
- **R or RAS:** Uses a **RAS** (*Remote Access Server*) Interface. A Driver connects to a RAS device using the **Ethernet** Interface and then sends an **AT** (*dial*) command

## Statistical Configuration

This section contains information about the configuration of **I/O Tags** and **Properties** of I/O Interfaces statistics.

### I/O Tags

#### Tags of I/O Interface Statistics (N2/B2 = 0)

The Tags described next display statistics for all I/O Interfaces.

#### IO.Stats.Partial.BytesRecv

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	0 (zero)
<b>N4 Parameter</b>	1101
<b>Configuration by String</b>	IO.Stats.Partial.BytesRecv

This Tag returns the number of bytes received in the current connection.

#### IO.Stats.Partial.BytesSent

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	0 (zero)
<b>N4 Parameter</b>	1100
<b>Configuration by String</b>	IO.Stats.Partial.BytesSent

This Tag returns the number of bytes sent through the current connection.

## IO.Stats.Partial.TimeConnectedSeconds

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	0 (zero)
<b>N4 Parameter</b>	1102
<b>Configuration by String</b>	IO.Stats.Partial.TimeConnectedSeconds

This Tag returns the number of seconds a Driver is connected in the current connection or 0 (zero) if a Driver is disconnected.

## IO.Stats.Partial.TimeDisconnectedSeconds

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	0 (zero)
<b>N4 Parameter</b>	1103
<b>Configuration by String</b>	IO.Stats.Partial.TimeDisconnectedSeconds

This Tag returns the number of seconds a Driver is disconnected since the last connection ended or 0 (zero) if a Driver is connected.

## IO.Stats.Total.BytesRecv

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	0 (zero)
<b>N4 Parameter</b>	1001
<b>Configuration by String</b>	IO.Stats.Total.BytesRecv

This Tag returns the number of bytes received since a Driver was loaded.

## IO.Stats.Total.BytesSent

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	0 (zero)
<b>N4 Parameter</b>	1000
<b>Configuration by String</b>	IO.Stats.Total.BytesSent

This Tag returns the number of bytes sent since a Driver was loaded.

## IO.Stats.Total.ConnectionCount

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	0 (zero)
<b>N4 Parameter</b>	1004
<b>Configuration by String</b>	IO.Stats.Total.ConnectionCount

This Tag returns the number of connections a Driver already established, successfully, since it was loaded.

## IO.Stats.Total.TimeConnectedSeconds

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	0 (zero)
<b>N4 Parameter</b>	1002
<b>Configuration by String</b>	IO.Stats.Total.TimeConnectedSeconds

This Tag returns the number of seconds a Driver remained connected since it was loaded.

## IO.Stats.Total.TimeDisconnectedSeconds

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1003
Configuration by String	IO.Stats.Total.TimeDisconnectedSeconds

This Tag returns the number of seconds a Driver remained disconnected since it was loaded.

## Properties

Currently, there are no properties defined specifically to display I/O Interface statistics at run time.

## Ethernet Interface Configuration

This section contains information about the configuration of **I/O Tags** and **Properties** of an **Ethernet** Interface.

## I/O Tags

### Tags of an Ethernet Interface (N2/B2 = 4)

The Tags described next allow controlling and identifying an **Ethernet** Interface at run time and they are also valid when the **RAS** Interface is selected.

#### IMPORTANT

These Tags are available **ONLY** while a Driver is in **Online** mode.

## IO.Ethernet.IPSelect

Type of Tag	I/O Tag
Type of Access	Reading or Writing
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	4 (four)
N4 Parameter	0 (zero)
String Configuration	IO.Ethernet.IPSelect

Indicates the active IP address. Possible values are **0**: The main IP address is selected, **1**: The first alternative or backup IP address is selected, **2**: The second alternative or backup IP address is selected, or **3**: The third alternative or backup IP address is selected.

If the **Ethernet** or **RAS** Interface is connected, this Tag indicates which one of the four configured IP addresses is in use. If the Interface is disconnected, this Tag indicates which IP address is used first on the next attempt to connect.

During the connection process, if the active IP address is not available, the I/O Interface tries to connect using the other IP address. If the connection with the alternative IP address works, it is configured as the active IP address (automatic switchover).

To force a manual switchover, write values from 0 (zero) to 3 (three) to this Tag. This forces a reconnection with the specified IP address (**0**: Main address or **1, 2, 3**: Alternative address) if a Driver is currently connected. If a Driver is disconnected, this Tag configures the active IP address for the next attempt to connect.

## IO.Ethernet.IPSwitch

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Write-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	4 (four)
<b>N4 Parameter</b>	1 (one)
<b>String Configuration</b>	IO.Ethernet.IPSwitch

Any value written to this Tag forces a manual switchover. If the main IP address is active, then the first alternative or backup IP address is activated, and so on for all alternative IP addresses and returning to the main address until a connection is established.

If a Driver is disconnected, this Tag configures the active IP address for the next attempt to connect.

## IO.Ethernet.SocketState

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	4 (four)
<b>N4 Parameter</b>	2 (two)
<b>String Configuration</b>	IO.Ethernet.SocketState

The Value property of this Tag corresponds to socket states as a map of bits:

- **Bit 0**: 0 (zero, not listening) or 1 (one, listening)
- **Bit 1**: 0 (zero, disconnected) or 1 (one, connected)

## Properties

These properties control the configuration of an **Ethernet** Interface.

**NOTE**

The **Ethernet** Interface is also used by the **RAS** Interface.

**IO.Ethernet.AcceptConnection**

☑ Configure to False if a Driver must not accept external connections, that is, if a Driver behaves as a master, or configure to True to enable the reception of connections, that is, if a Driver behaves as a slave.

**IO.Ethernet.BackupEnable[2,3]**

☑ Configure to True to enable an alternative or backup IP address. If the reconnection attempt with the main IP address fails, a Driver tries to use an alternative IP address. Configure to False to disable its usage.

**IO.Ethernet.BackupIP[2,3]**

📌 Alternative or backup IP address of a remote device. Users can use a numerical address, as well as a device's host name, such as "192.168.0.7" or "SERVER2".

**IO.Ethernet.BackupLocalPort[2,3]**

📌 Local port number to be used when connecting to an alternative IP address of a remote device. Used only if **IO.Ethernet.BackupLocalPortEnable** is equal to True.

**IO.Ethernet.BackupLocalPortEnable[2,3]**

☑ Configure to True to force the use of a specific local port when connecting to an alternative or backup IP address or configure to False to use any available local port.

**IO.Ethernet.BackupPort[2,3]**

📌 Port number of an alternative or backup IP address of a remote device, used with the **IO.Ethernet.BackupIP** property.

**IO.Ethernet.IPFilter**

📌 List with a comma-separated IPv4 or IPv6 addresses, which defines from which addresses a Driver accepts or blocks connections. Users can use asterisks, such as "192.168.\*.\*", or intervals, such as "192.168.0.41-50", in any part of IP addresses. To block an IP address or a range of IP addresses, use the tilde ("~") character at the beginning of the address, according to the next examples:

- **192.168.0.24**: Accepts only connections from IPv4 address 192.168.0.24
- **192.168.0.41-50**: Accepts connections from IPv4 addresses in the interval between 192.168.0.41 and 192.168.0.50
- **192.168.0.\***: Accepts connections from IPv4 addresses in the interval between 192.168.0.0 and 192.168.0.255
- **fe80:3bf:877:::\* (expands to fe80:03bf:0877:0000:0000:0000:0000:0000:\*)**: Accepts connections from IPv6 addresses in the interval between fe80:03bf:0877:0000:0000:0000:0000:0000 and fe80:03bf:0877:0000:0000:0000:ffff:ffff
- **192.168.0.10, 192.168.0.15, 192.168.0.20**: Accepts connections from IPv4 addresses 192.168.0.10, 192.168.0.15, and 192.168.0.20
- **~192.168.0.95, 192.168.0.\***: Accepts connections from IPv4 addresses in the interval between 192.168.0.0 and 192.168.0.255, except the IPv4 address 192.168.0.95

When a Driver receives a connection attempt, the list of filters is scanned sequentially from left to right, searching for a specific authorization or block for the IP address where the connection comes from. If no element on the list corresponds to the IP address, the authorization or block are dictated by the last element of that list:

- If the last element on the list is an authorization, such as "192.168.0.24", then all IP addresses not found on the list are blocked
- If the last element on the list is a block, such as "~192.168.0.24", then all IP addresses not found on the list are authorized

If an IP address appears on more than one filter on the list, the leftmost filter has precedence. For example, in case of "~192.168.0.95, 192.168.0.\*", the IP address 192.168.0.95 fits both rules, but the rule that wins is the leftmost one, "~192.168.0.95", and therefore this IP address is blocked.

When **IOKit** blocks a connection, it logs a message "Blocked incoming socket connection from {IP}!".

In case of UDP connections in broadcast listening mode, in which a Driver can receive packets from different IP addresses, blocks or permissions are performed at each packet received. If a packet is received from a blocked IP address, it logs a message "Blocked incoming packet from {IP} (discarding {N} bytes)!".

## IO.Ethernet.ListenIP

**A** IP address of the local network interface that a Driver uses to establish and accept connections. Leave this property empty to establish and accepts connections using any local network interface.

## IO.Ethernet.ListenPort

**9** Number of the IP port used by a Driver to listen to connections.

## IO.Ethernet.MainIP

**A** IP address of a remote device. Users can use a numerical address, as well as a device's host name, such as "192.168.0.7" or "SERVER2".

## IO.Ethernet.MainLocalPort

**9** Local port number to use when connecting to the main IP address of a remote device. This value is only used if the **IO.Ethernet.MainLocalPortEnable** property is equal to True.

## IO.Ethernet.MainLocalPortEnable

Configure to True to force the use of a specific local port when connecting to the main IP address of a remote device or configure to False to use any available local port.

## IO.Ethernet.MainPort

**9** Number of the IP port of a remote device, used with the **IO.Ethernet.MainIP** property.

## IO.Ethernet.PingEnable

Configure to True to enable sending a **ping** command to the IP address of a remote device, before trying to connect to the socket. This socket's connection time-out cannot be controlled, therefore sending a **ping** command before connecting is a fast way to detect if the connection is going to fail. Configure to False to disable a **ping** command.

## IO.Ethernet.PingTimeoutMs

9 Delay time to wait for a response from a **ping** command, in milliseconds.

## IO.Ethernet.PingTries

9 Maximum number of attempts of a **ping** command. Minimum value is 1 (one), including the first **ping** command.

## IO.Ethernet.ShareListenPort

☑ Configure to True to share a listening port with other Drivers and processes or False to open a listening port in exclusive mode. To successfully share a listening port, all Drivers and processes that use that port must open it in shared mode. When a listening port is shared, each incoming connection is distributed to one of the processes listening. This way, if a Slave Driver only supports one connection at a time, users can use several instances of this Driver listening on the same port, therefore simulating a Driver with support for multiple connections.

## IO.Ethernet.SupressEcho

☑ Configure to True to eliminate echoes in communication. An echo is the unwanted reception of an exact copy of all data packets a Driver sent to a device.

## IO.Ethernet.Transport

A Defines a transport protocol. Possible values are **T or TCP**: Uses the TCP/IP protocol or **U or UDP**: Uses the UDP/IP protocol.

## IO.Ethernet.UseIPv6

☑ Configure to True to use IPv6 addresses on all Ethernet connections or configure to False to use IPv4 addresses (default).

# Modem Interface Configuration

This section contains information about the configuration of **I/O Tags** and **Properties** of a **Modem** (TAPI) Interface.

## I/O Tags

### Tags of a Modem Interface (N2/B2 = 3)

The Tags described next allow controlling and diagnosing a **Modem** (TAPI) Interface at run time.

#### IMPORTANT

These Tags are available **ONLY** while a Driver is in **Online** mode.

## IO.TAPI.ConnectionBaudRate

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	3 (three)
<b>N4 Parameter</b>	5 (five)
<b>String Configuration</b>	IO.TAPI.ConnectionBaudRate

Indicates a baud rate value for the current connection. If a modem is not connected, returns the value 0 (zero).

## IO.TAPI.Dial

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Write-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	3 (three)
<b>N4 Parameter</b>	1 (one)
<b>String Configuration</b>	IO.TAPI.Dial

Write any value to this Tag to force a **Modem** Interface to start a call. This is an asynchronous command, only starting the call process. Users can monitor the **IO.TAPI.IsModemConnected** Tag to detect when a call is established.

## IO.TAPI.HangUp

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Write-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	3 (three)
<b>N4 Parameter</b>	4 (four)
<b>String Configuration</b>	IO.TAPI.HangUp

Any value written to this Tag hangs the current call up.

**NOTE**

Use this command only when managing the physical layer manually or when explicitly trying to force a Driver to restart the communication. If the physical layer is configured for automatic reconnection, a Driver immediately tries to reestablish the connection.

**IO.TAPI.IsModemConnected**

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	3 (three)
<b>N4 Parameter</b>	3 (three)
<b>String Configuration</b>	IO.TAPI.IsModemConnected

This Tag indicates the status of a modem connection. Possible values are **0**: The modem is not connected, but it may be performing or receiving an external call or **1**: The modem is connected and a Driver completed or received an external call successfully. While it is in this status, the physical layer can send or receive data.

**IO.TAPI.IsModemConnecting**

<b>Type of Tag</b>	I/O Tag
<b>Type of Access</b>	Read-Only
<b>N1 Parameter</b>	-1 (minus one)
<b>N2 Parameter</b>	0 (zero)
<b>N3 Parameter</b>	3 (three)
<b>N4 Parameter</b>	6 (six)
<b>String Configuration</b>	IO.TAPI.IsModemConnecting

This Tag indicates the status of a modem connection, with more details than the **IO.TAPI.IsModemConnected** Tag. Possible values are **0**: Modem is not connected, **1**: Modem is connecting, that is, performing or receiving an external call, **2**: Modem is connected. While in this status, the physical layer can send or receive data, or **3**: Modem is disconnecting the current call.

## IO.TAPI.ModemStatus

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	2 (two)
String Configuration	IO.TAPI.ModemStatus

Returns a **String** with the current status of a modem. Possible values are the following:

- **"No status!"**: The **Modem** Interface was not open yet or was already closed
- **"Modem initialized OK!"**: The **Modem** Interface was initialized successfully
- **"Modem error at initialization!"**: A Driver could not initialize modem's line. Check that Driver's log file for more details
- **"Modem error at dial!"**: A Driver could not start or accept a call
- **"Connecting..."**: A Driver started a call successfully, and is currently processing that call
- **"Ringing..."**: Indicates that the modem is receiving an external call, but it did not accepted it yet
- **"Connected!"**: A Driver connected successfully, that is, completed or accepted an external call
- **"Disconnecting..."**: A Driver is turning the current call off
- **"Disconnected OK!"**: A Driver turned the current call off
- **"Error: no dial tone!"**: A Driver aborted a call because the available line signal was not detected
- **"Error: busy!"**: A Driver aborted a call because the line was busy
- **"Error: no answer!"**: A Driver aborted a call because no answer was received from the other modem
- **"Error: unknown!"**: Current call was aborted because of an unknown error

## IO.TAPI.PhoneNumber

Type of Tag	I/O Tag
Type of Access	Reading or Writing
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	0 (zero)
String Configuration	IO.TAPI.PhoneNumber

This Tag is a **String** that reads or changes the telephone number used by the **IO.TAPI.Dial** Tag. When changing this Tag, the new value is used only on the next **Dial** command.

## Properties

These properties control the configuration of a **Modem** (TAPI) Interface.

### IO.TAPI.AcceptIncoming

**9** Configure to False if a modem cannot accept external calls, that is, if a Driver behaves as a master, and configure to True to enable receiving calls, that is, if a Driver behaves as a slave.

### IO.TAPI.ModemID

**9** This is the identification number of a modem. This ID is created by Windows and used internally to identify a modem on a list of devices installed on a computer. This ID may not remain valid if a modem is reinstalled or an application is executed on another computer.

#### NOTE

It is advisable to configure this property as 0 (zero), indicating that a Driver must use the first available modem.

### IO.TAPI.PhoneNumber

**A** A telephone number used by **Dial** commands, such as "0w01234566", in which the "w" character forces a modem to wait for a call sign.

## RAS Interface Configuration

This section contains information about the configuration of **I/O Tags** and **Properties** of a **RAS** Interface.

### I/O Tags

#### Tags of a RAS Interface (N2/B2 = 5)

Currently, there are no Tags defined specifically to manage a **RAS** Interface at run time.

### Properties

These properties control the configuration of a **RAS** Interface.

#### NOTE

A **RAS** Interface uses the **Ethernet** Interface, which therefore must be also configured.

### IO.RAS.ATCommand

**A** An **AT** command to send through a socket to force a RAS device to perform a call using the current RAS channel, such as "ATDT6265545".

## IO.RAS.CommandTimeoutSec

9 Time to wait for a **CONNECT** message in response to an **AT** command, in seconds.

## Serial Interface Configuration

This section contains information about the configuration of **I/O Tags** and **Properties** of a **Serial** Interface.

### I/O Tags

#### Tags of a Serial Interface (N2/B2 = 2)

Currently, there are no Tags defined specifically to manage a **Serial** Interface at run time.

### Properties

These properties control the configuration of a **Serial** Interface.

#### IO.Serial.Baudrate

9 Specifies a baud rate of a serial port, such as 9600.

#### IO.Serial.CTSTimeoutMs

9 Time to wait for a **CTS** signal, in milliseconds. After turning the **RTS** signal on, a timer is started to wait for a **CTS** signal. If this timer expires, a Driver aborts sending bytes through the serial port. Available only when the **IO.Serial.RTS** property is configured with the value **Toggle** and the **IO.Serial.WaitCTS** property is configured to True.

#### IO.Serial.DataBits

9 Specifies the number of data bits to configure a serial port. Possible values are **5**: Five data bits, **6**: Six data bits, **7**: Seven data bits, or **8**: Eight data bits.

#### IO.Serial.DelayAfterMs

9 Number of milliseconds to delay after the last byte is sent through a serial port, but before turning the **RTS** signal off. Available only when the **IO.Serial.RTS** property is configured with the value **Toggle** and the **IO.Serial.WaitCTS** property is configured to False.

#### IO.Serial.DelayBeforeMs

9 Number of milliseconds to delay after turning the **RTS** signal on, but before data is sent. Available only when the **IO.Serial.RTS** property is configured with the value **Toggle** and the **IO.Serial.WaitCTS** property is configured to False.

#### IO.Serial.DTR

A Indicates how a Driver deals with the **DTR** signal. Possible values are **OFF**: **DTR** signal is always turned off or **ON**: **DTR** signal is always turned on.

#### IO.Serial.InterbyteDelayUs

9 Delay time, in milliseconds (1/1000000 of a second), for each two bytes sent through a **Serial** Interface.

## IO.Serial.InterframeDelayMs

**9** Delay time, in milliseconds, before sending a packet after the last packet sent or received.

## IO.Serial.Parity

**A** Specifies a parity for the configuration of a serial port. Possible values are **E or Even**: Even parity, **N or None**: No parity, **O or Odd**: Odd parity, **M or Mark**: Mark parity, or **S or Space**: Space parity.

## IO.Serial.Port

**9** Number of the local serial port. Possible values are **1**: Uses the COM1 port, **2**: Uses the COM2 port, **3**: Uses the COM3 port, or **n**: Uses the COMn port.

## IO.Serial.RTS

**A** Indicates how a Driver deals with the **RTS** signal. Possible values are **OFF**: **RTS** signal always off, **ON**: **RTS** signal always on, or **Toggle**: Turns the **RTS** signal on when transmitting data and turns the **RTS** signal off when not transmitting data.

## IO.Serial.StopBits

**9** Specifies the number of stop bits for the configuration of a serial port. Possible values are **1**: One stop bit, **2**: One and a half stop bit, or **3**: Two stop bits.

## IO.Serial.SuppressEcho

**9** Use a value different from 0 (zero) to enable suppressing the echo or 0 (zero) to disable it.

## IO.Serial.WaitCTS

**▣** Configure to True to force a Driver to wait for the **CTS** signal before sending bytes when the **RTS** signal is turned on. Available only when the **IO.Serial.RTS** property is configured with the value **Toggle**.

## Driver's Revision History

VERSION	DATE	AUTHOR	COMMENTS
2.0.8	04/02/2026	M. Ludwig	<ul style="list-style-type: none"> <li>Driver updated to <b>IOKit</b> library version <b>3.0</b> and Visual Studio 2022 (Case 38002).</li> </ul>
2.0.7	03/21/2023	M. Ludwig	<ul style="list-style-type: none"> <li>Implemented a feature of ignoring errors considered non-fatal in a Unit (Case 33910).</li> </ul>
2.0.6	08/02/2021	M. Ludwig	<ul style="list-style-type: none"> <li>Fixed a communication error when Tags with a <b>String</b> data type are configured with a length above 15 (Case 31178).</li> </ul>
2.0.5	08/05/2019	M. Ludwig	<ul style="list-style-type: none"> <li>Driver ported to Visual Studio 2017 (Case 27088).</li> </ul>

VERSION	DATE	AUTHOR	COMMENTS
2.0.4	03/27/2019	M. Ludwig	<ul style="list-style-type: none"> <li>Expanded the number of destination nodes for up to 255 (<i>Case 26383</i>).</li> </ul>
2.0.3	06/12/2017	M. Ludwig	<ul style="list-style-type: none"> <li>Added a <b>String</b> format with low-high byte order (<i>Case 22773</i>).</li> </ul>
2.0.2	08/29/2014	M. Ludwig	<ul style="list-style-type: none"> <li>Fixed a communication error when Tags with <b>Bit</b> registers are configured (<i>Case 16872</i>).</li> </ul>
2.0.1	10/28/2013	M. Ludwig	<ul style="list-style-type: none"> <li>Fixed data validation when the end code indicates an error communicating on a serial interface and Host Link protocol (<i>Case 14389</i>).</li> <li>Driver ported to <b>IOKit</b> library version <b>2.0</b> (<i>Case 14469</i>).</li> </ul>
1.7.1	07/01/2011	M. Ludwig	<ul style="list-style-type: none"> <li>Driver ported to <b>Windows CE</b> (<i>Case 12300</i>).</li> </ul>
1.6.1	01/26/2011	M. Ludwig	<ul style="list-style-type: none"> <li>Conformity for communication with more than one <b>Master</b> on a network (<i>Case 11895</i>).</li> </ul>
1.5.1	06/09/2009	M. Ludwig	<ul style="list-style-type: none"> <li>Fixed a problem when returning errors on successful writings of memory areas and other commands (<i>Case 10525</i>).</li> </ul>
1.4.1	01/28/2009	M. Ludwig	<ul style="list-style-type: none"> <li>Fixed a problem with a successful return of writings on unsuccessful situations (<i>Case 10165</i>).</li> <li>Added useful information to User's Manual (<i>Case 9739</i>).</li> </ul>
1.3.1	01/29/2008	M. Ludwig	<ul style="list-style-type: none"> <li>Fixed a problem with error responses on writings, even when successful (<i>Case 8990</i>).</li> <li>Driver updated to <b>IOKit</b> library version <b>1.14</b> (<i>Case 9014</i>).</li> </ul>
1.2.1	11/04/2005	M. Ludwig	<ul style="list-style-type: none"> <li>Fixed a problem with swapped <b>Words</b> and a problem when writing blocks to wrong memory locations (<i>Case 5947</i>).</li> </ul>
1.1.1	01/14/2005	M. Salvador	<ul style="list-style-type: none"> <li>Added support for UDP protocol.</li> </ul>
1.0.1	08/23/2004	M. Ludwig	<ul style="list-style-type: none"> <li>Initial version of this Driver.</li> </ul>

**Headquarters**

**Rua Mostardeiro, 322/Cj. 902, 1001 e  
1002**

**90510-002 — Porto Alegre — RS**

**Phone: (+55 51) 3346-4699**

**Fax: (+55 51) 3222-6226**

**E-mail: [elipse-rs@elipse.com.br](mailto:elipse-rs@elipse.com.br)**

**Branch in Taiwan**

**9F., No.12, Beiping 2nd St., Sanmin Dist.**

**807 — Kaohsiung City — Taiwan**

**Phone: (+886 7) 323-8468**

**Fax: (+886 7) 323-9656**

**E-mail: [evan@elipse.com.br](mailto:evan@elipse.com.br)**

**Check our website for information about a representative in your country.**

**[www.elipse.com.br](http://www.elipse.com.br)**

**[kb.elipse.com.br](http://kb.elipse.com.br)**

**[forum.elipse.com.br](http://forum.elipse.com.br)**

**[www.youtube.com/elipsesoftware](http://www.youtube.com/elipsesoftware)**

**[elipse@elipse.com.br](mailto:elipse@elipse.com.br)**



Gartner, Cool Vendors in Brazil 2014, April 2014.

Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability of fitness for a particular purpose.

**Microsoft Partner**  
Gold Independent Software Vendor (ISV)