ABB SPABus32 Driver

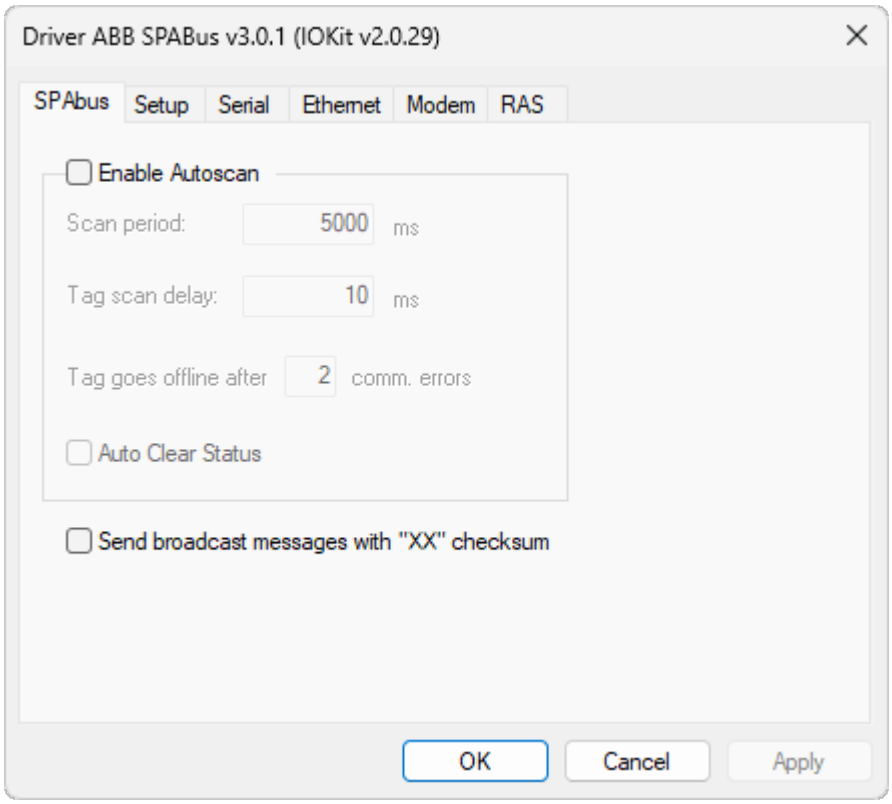
File Name	SPABus32.dll
Manufacturer	Asea Brown Bowery Ltd.
Devices	REF/RET/REM 541/543/545 Feeder Terminals, REX521 Relay, and REJ/REU 521/525/527 Relays
Protocol	SPAbus Communication Protocol version 2.5
Version	3.0.2
Last Update	08/22/2025
Platform	Win32
Dependencies	None
Superblock Readings	No
Level	0

Introduction

This is the ABB SPABus32 Driver, which allows communication between **Elipse Software** systems and ABB product series compatible with the **SPABus** protocol.

Configuring the Driver

Starting with version **2.0**, this Driver does not use the **[P]** configuration parameters anymore. All settings must be performed on this Driver's configuration window. The configuration of communication must be performed using the configuration tabs of **IOKit** library. For more information, please check topic **Documentation of I/O Interfaces**. The **SPAbus** tab is shown on the next figure.



SPAbus tab

The available options on this tab are described on the next table.

Available options on the SPAbus tab

OPTION	DESCRIPTION
Enable Autoscan	Enables an automatic scanning for events from **L** and **B** categories. For more information, please check topic **Automatic Scanning of Events**
Scan period	Defines the scan period, in milliseconds, for the automatic scanning of events. For more information, please check topic **Automatic Scanning of Events**
Tag scan delay	Defines a time delay, in milliseconds, between the scanning of each Tag registered for automatic scanning, during this Driver's internal scan, to prevent overloading an application
Tag goes offline after *n* comm. errors	Defines the number of communication errors after which an automatic scanning of events Tag is considered in an **Offline** status. This Tag returns to an **Online** status as soon as any communication is successful
Auto Clear Status	If this option is enabled during an automatic scanning of events, if this Driver receives events with codes **E50** or **E51**, the internal thread for scanning automatically sends a command to clear the status of a device, that is, writing 0 (zero) to the **C** category, removing a slave from an error condition. Collected **E50** and **E51** events are returned normally to an application
Send broadcast messages with "XX" checksum	This option enables sending and receiving frames with the "XX" characters replacing the checksum. This is usually applied for testing, but it is recommended to disable this option, which is its default value, in production, because

OPTION	DESCRIPTION
	without a checksum invalid frames can be received, from noise, which may lead to an invalid reading of values

[N] Parameters for Addressing PLC Tags

N1	Number of a slave. Use the value 900 for a broadcast address
N2	Number of a channel. The default value is 0 (zero)
N3	Data type. For more information, please check the next table
N4	Number of a data by the category of a slave (I, O, S, V, or M)

To communicate with REJ-525 relays, add the value 5000 to the *N1* parameter. For example, to address the REJ-525 slave with number 3 (three), configure the *N1* parameter with the value 5003.

The *N3* parameter must have one of the values referring to the type of function or data type to be executed, read, or written, as described on the next table. For data types **17** and **19**, users can configure the *N4* parameter with an additional time, in milliseconds, to compensate a network delay. The syncing interval is defined by a Tag's scan time.

Data types retrieved by PLC Tags

VALUE	DESCRIPTION
0	Data input (I)
1	Data output (O)
2	Configuration of values (S)
3	Internal variables (V)
4	Memory data (M)
5	Status of a slave (C)
6	Identification of a slave (F)
7	Hour (D)
8	Last events (L) in **Text** format
9	Data and time (T)
10	Last events from the backup buffer (B) in **Text** format
11	Valid alarm (A)
17	Clock syncing Tag (full date and time, read-only)
19	Clock syncing Tag (only seconds, read-only)
101	Indicates whether there are events stored in this Driver's internal buffer of events
102	Indicates whether the respective Tag for automatic scanning of events is in **Online** mode

[B] Parameters for Addressing Block Tags

This Driver allows reading data from Blocks or from Block Elements and writing to Block Elements. Users cannot write to full Blocks. The writing must be always performed by Elements.

B1	Number of a slave number. Use the value 900 for a broadcast address
B2	Number of a channel. The default value is 0 (zero)
B3	Category of data or function type. For more information, please check the next table
B4	Number of the index of the first Element of a Block Tag

To communicate with REJ-525 relays, add the value 5000 to the *B1* parameter. For example, to address the slave REJ-525 with number 3 (three), configure the *B1* parameter with the value 5003.

Categories of data retrieved by Block Tags

VALUE	DESCRIPTION
0	Data input (I)
1	Data output (O)
2	Configuration of values (S)
3	Internal variables (V)
4	Memory data (M)
8	Last events (L). This option uses a Block with the Elements **Date and Time**, **Number of a Channel**, and **Code of an Occurrence's Cause**
10	Last events from the backup buffer (B). This option uses a Block with the Elements **Date and Time**, **Number of a Channel**, and **Code of an Occurrence's Cause**
13	Reads conditions from events in internal variables (V), with the first Element transformed into date and time
100	Tag for automatic scanning of events

Reference for PLC Tags

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

Data Input

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	Channel
N3	0 (zero)
N4	Number of a data

Corresponds to the **I** category. Allows reading or writing analog data from a slave, as well as digital input statuses. Data type of the **Value** property is a floating point.

Data Output

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	Channel
N3	1 (one)
N4	Number of a data

Corresponds to the **O** category. Allows reading or writing analog data or digital output statuses. Data type of the **Value** property is a floating point.

Configuration of Values

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	Channel
N3	2 (two)
N4	Number of a data

Corresponds to the **S** category. Allows reading or writing parameters from a slave. Data type of the **Value** property is a floating point.

Internal Variables

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	Channel
N3	3 (three)
N4	Number of a data

Corresponds to the **V** category. Allows reading and writing internal variables of a device. Data type of the **Value** property is a floating point. The values of internal variables provide access to data or functions, such as:

- Additional data on the supervised process and the respective events
- Additional data related to a slave or its functionality
- Control over functions from an operating slave
- Programming general functions in a slave

Memory Data

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	Channel
N3	4 (four)
N4	Number of a data

Corresponds to the **M** category. Allows reading or writing data to a slave's memory. It may include data from metering or status stored in a device's memory. Data type of the **Value** property is a **String**.

Status of a Slave

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	0 (zero)
N3	5 (five)
N4	0 (zero)

Corresponds to the **C** category. The status of a slave is represented using 2 (two) bits, in which the returned message may contain values 0 (zero), 1 (one), 2 (two), or 3 (three). Data type of the **Value** property is an unsigned integer. The meaning of each bit is the following:

- **Bit 0**: Reset of a slave or another situation that may have caused a loss of event data. When this bit is turned on, that is, C is equal to 1 (one) or 3 (three), a slave sends, as a response to the reading of events, the **E50** event until this bit is cleared, by writing 0 (zero) to this Tag, that is, C is equal to 0 (zero)
- **Bit 1**: Indicates an overflow in the buffer of events of a slave. When this bit is active, that is, C is equal to 2 (two) or 3 (three), a slave always sends the **E51** event as a response to the reading of events, until this bit is cleared, by writing 0 (zero) to this Tag, that is, C is equal to 0 (zero). During this overflow, a slave usually stops adding new events to the buffer, and the status must be cleared to restart storing events

Identification of a Slave

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	0 (zero)
N3	6 (six)
N4	0 (zero)

Corresponds to the **F** category. Allows access to an identification code of the type of a slave. This code can be, for example, the product identification of a slave, or product code. Data type of the **Value** property is a **String**.

Time

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	0 (zero)
N3	7 (seven)
N4	0 (zero)

Corresponds to the **D** category. Allows reading and writing date and time. Data type of the **Value** property is a **Date and Time**.

Last Events

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	0 (zero)
N3	8 (eight)
N4	0 (zero)

Corresponds to the **L** category. Allows reading the most recent events from the buffer of a slave. When events are read as a PLC Tag, they are returned as a **String** in the format **<timestamp in seconds><channel>E<event code>**. In case of an error, to retry the reading of an event, users must read the backup buffer. For more information, please check the **Last Event of the Backup Buffer** PLC Tag and the **Last Events of the Backup Buffer** Block Tag. Data type of the **Value** property is a **String**. The **Timestamp** property of this Block Tag contains the timestamp returned by a device, which corresponds to the first field of the **String** from the **Value** property.

Date and Time

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	0 (zero)
N3	9 (nine)
N4	0 (zero)

Corresponds to the **T** category. Allows reading and writing time values in seconds. Data type of the **Value** property is a floating point.

Last Event of the Backup Buffer

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	0 (zero)
N3	10
N4	4 (four)

Corresponds to the **B** category. It allows reading again a lost event after a communication error when reading an **L** category. For more information, please check the **Last Events** Tag. When an event is read as a PLC Tag, the identification **String** of that event is returned as a **String** in the format **<timestamp in seconds><channel>E<event code>**. Data type of the **Value** property is a **String**. The **Timestamp** property of this Tag contains the timestamp returned by a device, which corresponds to the first field of the **String** from the **Value** property.

Valid Alarm

Reading and Writing

N1	Address of a slave. Use the value 900 for a broadcast address
N2	0 (zero)
N3	11
N4	0 (zero)

Corresponds to the **A** category. During the reading of this Tag, a slave responds with events from valid alarms without time information. Data type of the **Value** property is a **String**.

Clock Syncing: Date and Time

Read-Only

N1	Address of a slave. Use the value 900 for a broadcast address
N2	0 (zero)
N3	17
N4	Additional time, in milliseconds

Every reading operation in this Tag sends a clock syncing command to a slave, syncing that slave's internal clock with the computer's clock. The category used is **D**. To compensate any network delay when sending a command, users can define an additional time to add to the computer's time in the *N4* parameter. The **Value** property is not used and returns 0 (zero) in **Broadcast** mode.

Clock Syncing: Only Seconds

Read-Only

N1	Address of a slave. Use the value 900 for a broadcast address
N2	0 (zero)
N3	19
N4	Additional time, in milliseconds

Every reading operation in this Tag sends a clock syncing command to a slave, syncing that slave's internal clock with the computer's internal clock. The category used is **T**. What differentiates this Tag from the **Clock Syncing: Date and Time** Tag is the category of data, **T** instead of **D**, and the format of data sent to a device, only in seconds. Their functionality is the same. To compensate any network delay when sending a command, users can define an additional time to the computer's time in the *N4* parameter. The **Value** property is not used and returns 0 (zero) in **Broadcast** mode.

Events in Buffer

Read-Only

N1	0 (zero)
N2	0 (zero)
N3	101
N4	0 (zero)

This Tag indicates if there are events in the buffer not yet read by an application, collected using the internal scan of this Driver. This Tag can be used to prevent loosing events when closing an application. By setting this Driver to the **Offline** mode, the internal scan is suspended, allowing the Tags for automatic scanning of events to remove data from the internal buffer, until this Tag assumes the value 0 (zero). After emptying this Driver's internal buffer, the I/O Driver object can be destroyed or stopped, without loosing events. For more information, please check topic **Automatic Scanning of Events**. Possible values for the **Value** property are **0**: There is no data in the buffer or **1**: There is still data in the buffer waiting for reading by an application.

Online Event

Read-Only

N1	Address of a slave. Use the value 900 for a broadcast address
N2	0 (zero)
N3	102
N4	0 (zero)

This Tag is related to the **Automatic Scanning of Events** and indicates if that Block Tag is in **Online** or **Offline** mode, allowing a diagnostics on the connection with a slave. That Block Tag goes to the **Online** mode whenever the communication with a device is successful, and it goes to the **Offline** mode after a certain number of communication errors. This value can be configured in the **Tag goes offline after n comm. errors** option. For more information, please check topic **Automatic Scanning of Events**. Possible values for the **Value** property are **0**: The **Automatic Scanning of Events** Block Tag is in **Offline** mode or **1**: The **Automatic Scanning of Events** Block Tag is in **Online** mode.

Reference for Block Tags

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

Data Input

Reading and Writing

B1	Address of a slave. Use the value 900 for a broadcast address
B2	Channel
B3	0 (zero)
B4	Number of a data

Corresponds to the **I** category. Allows reading or writing analog data from a slave, as well as digital input statuses. Data type of the **Value** property of Block Elements is a floating point.

Data Output

Reading and Writing

B1	Address of a slave. Use the value 900 for a broadcast address
B2	Channel
B3	1 (one)
B4	Number of a data

Corresponds to the **O** category. Allows reading or writing analog data or digital output statuses. Data type of the **Value** property of Block Elements is a floating point.

Configuration of Values

Reading and Writing

B1	Address of a slave. Use the value 900 for a broadcast address
B2	Channel
B3	2 (two)
B4	Number of a data

Corresponds to the **S** category. Allows reading or writing parameters of a slave. Data type of the **Value** property of Block Elements is a floating point.

Internal Variables

Reading and Writing

B1	Address of a slave. Use the value 900 for a broadcast address
B2	Channel
B3	3 (three)
B4	Number of a data

Corresponds to the **V** category. Allows reading or writing internal variables of a device. Data type of the **Value** property of Block Elements is a floating point. The values of internal variables provide access to data or functions, such as:

- Additional data on the supervised process and the respective events
- Additional data related to a slave or its functionality
- Control over functions from an operating slave
- Programming general functions in a slave

Memory Data

Reading and Writing

B1	Address of a slave. Use the value 900 for a broadcast address
B2	Channel
B3	4 (four)
B4	Number of a data

Corresponds to the **M** category. Allows reading or writing data to the memory of a slave. It may include data from metering or status stored in the memory of a device. Data type of the **Value** property of Block Elements is a **String**.

Last Events

Reading and Writing

B1	Address of a slave. Use the value 900 for a broadcast address
B2	0 (zero)
B3	8 (eight)
B4	0 (zero)

Corresponds to the **L** category. Allows reading the most recent events from the buffer of a slave. When an event is read as a PLC Tag, that event is returned as a **String**. For more information, please check topic **Reference for PLC Tags**. When reading as a Block Tag, events are returned with the following Elements:

- **Element 0**: Timestamp
- **Element 1**: Channel
- **Element 2**: Code of an event

The **Timestamp** property returns the same value of Block Element 0 (zero).

NOTE

Analog events are not yet supported for reading as Block Tags. Notice that users can configure a slave by using an internal variable and thus not sending analog events. For more information, please check the documentation of the manufacturer.

Last Events of the Backup Buffer

Reading and Writing

B1	Address of a slave. Use the value 900 for a broadcast address
B2	0 (zero)
B3	10
B4	0 (zero)

Corresponds to the **B** category. Allows reading the most recent events of the buffer of a slave. When an event is read as a PLC Tag, this event is returned as a **String**. For more information, please check topic **Reference for PLC Tags**. When reading as a Block Tag, events are returned with the following Elements:

- **Element 0**: Timestamp
- **Element 1**: Channel
- **Element 2**: Code of an event

The **Timestamp** property returns the same value of Block Element 0 (zero).

Event Conditions

Read-Only

B1	Address of a slave. Use the value 900 for a broadcast address
B2	Channel
B3	13
B4	0 (zero)

Corresponds to the **V** category. It reads event conditions from internal variables of a slave into a Block Tag. The first Block Element is returned in **Date and Time** format. The other Block Elements are returned in 32-bit floating point format.

Automatic Scanning of Events

Reading and Writing

B1	Address of a slave. Use the value 900 for a broadcast address
B2	0 (zero)
B3	100
B4	0 (zero)

This Tag returns events stored in the internal buffer of this Driver, previously collected using the internal scan for automatic scanning of events. For more information, please check topic **Automatic Scanning of Events**. The Block Elements of this Block Tag are the following:

- **Element 0**: Timestamp
- **Element 1**: Channel
- **Element 2**: Code of an event

The **Timestamp** property returns the same value of Block Element 0 (zero).

Automatic Scanning of Events

Starting with version **2.0**, this Driver has a feature for automatically scanning events.

The functionality of automatic scanning uses an internal thread of this Driver to perform a scan for events of a device. Events collected from a device are stored in an internal buffer of this Driver and returned to an application using the Block Tag **Automatic Scanning of Events**.

The events read by this Block Tag are removed from the internal buffer of this Driver and reported by event, that is, each reading operation of this Block Tag triggers a series of **OnRead** events. For more information, please check topic *I/O Tags Reported by Events* on **Elipse E3 User's Manual**.

The automatic scanning of events is equivalent to the reading of Block Tags **Last Events** and **Last Events of the Backup Buffer**, that is, the **L** and **B** categories. Readings are always performed using Block Tags with 3 (three) Block Elements. This Driver starts its internal scan by reading the **L** category. If there is a communication error on a certain channel and slave,

that reading is automatically switched to the **B** category (backup) to avoid loosing events. After the first successful reading of events from the **B** category, this Driver then returns to search for events from the **L** category.

Just like when reading Block Tags **Last Events** and **Last Events of the Backup Buffer**, this Driver does not support analog events.

Configuration of Automatic Scanning

To use the automatic scanning of events, users must enable the automatic scanning on the configuration window of this Driver. Users must also configure the scan period and the scan delay among Tags. At every scanning period, this Driver starts scanning all Tags registered for automatic reading of events, with a delay between every reading. The delay among Tag readings allows balancing the internal scan load, thus avoiding to overload applications and communication channels at every scan.

The first reading operation of any **Automatic Scanning of Events** Block Tag does not return values, that is, returns an empty list, it is only intended to internally register that Block Tag in this Driver's automatic scanning mechanism. No property of that Block Tag is changed. Then, that Block Tag starts to be included in each automatic scanning of events. The next readings start to return the collected values, whenever they are available in the internal buffer of this Driver. Scan times of automatic scanning of events in an application can be low, such as 100 milliseconds. As these Tags do not generate communication with a device, their CPU consumption is low.

Auto Clear Status

In addition to capturing events, users can configure the thread for internal scanning to send a writing of a value 0 (zero) to the **C** category, that is, C is equal to 0 (zero), automatically, whenever receiving events with codes **E50** or **E51**. This removes a device from an error condition or overflow, reported by these events. The collected **E50** and **E51** events are stored in the internal buffer of this Driver and returned normally to an application. To use this feature, please enable the **Auto Clear Status** option on the configuration window.

Preventing a Loss of Events When Closing this Driver

The **Events in Buffer** PLC Tag allows monitoring the presence of data in the internal buffer of this Driver. This PLC Tag can be used to avoid loosing events when closing an application. If it is important to avoid loosing events that remain in the internal buffer when closing an application, users must halt the internal scan, by setting this Driver to the **Offline** mode before closing it, and leaving the automatic scanning of Tags enabled. Thus, new events are not added to the buffer while automatic scanning Tags process the remaining events. Wait for the **Events in Buffer** PLC Tag to return to 0 (zero) and then close this Driver safely, without loosing events.

Monitoring Communication Status with Every Slave

To monitor the status of the automatic scanning of Tags, as well as the status of communication with every slave, the **Online Event** PLC Tag was created. This Tag reference automatic scanning Tags using the *N1* and *N2* parameters, returning 0 (zero, **Offline** mode) or 1 (one, **Online** mode) to indicate the status of communication for each Tag. Automatic scanning Tags are considered in **Online** mode whenever an automatic scanning is successful, and are considered in **Offline** mode after a certain number of consecutive communication errors. The maximum number of communication errors allowed without a Tag being considered as in **Offline** mode can be configured in the **Tag goes offline after n comm. errors** option on the configuration window.

Documentation of I/O Interfaces

This section contains the documentation of I/O Interfaces referring to **SPABus32** 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 **Elipse 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 **Elipse E3** version 2.0 or later, click **Configure driver** on a Driver's toolbar. In **Elipse 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.

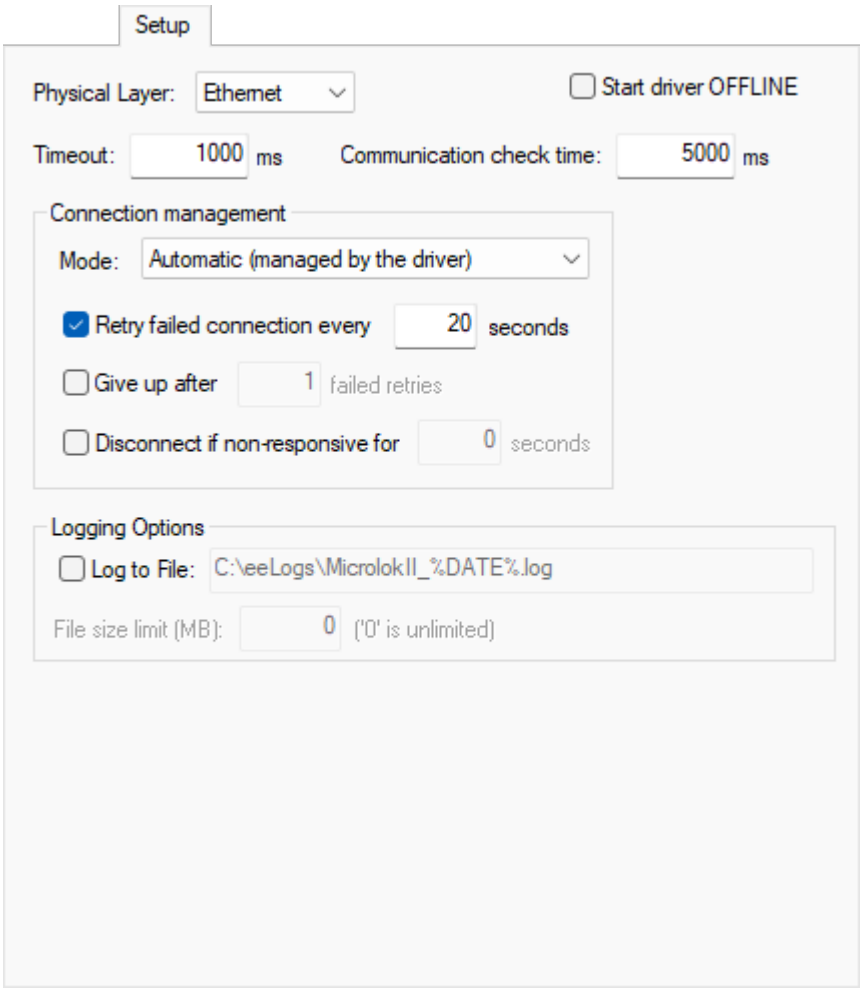
Configuration Dialog Box

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 tab

General options on the Setup tab

OPTION	DESCRIPTION
Physical Layer	Select the physical layer on a list. Available options are **Serial**, **Ethernet**, **Modem**, and **RAS**. The selected interface must be configured on its specific tab
Timeout	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
Communication check time	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 **IO.CommunicationStatus** Tag
Start driver OFFLINE	Select this option so that a Driver starts in **Offline** mode or stopped. This means that the I/O interface is not created until this Driver is configured to **Online** 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

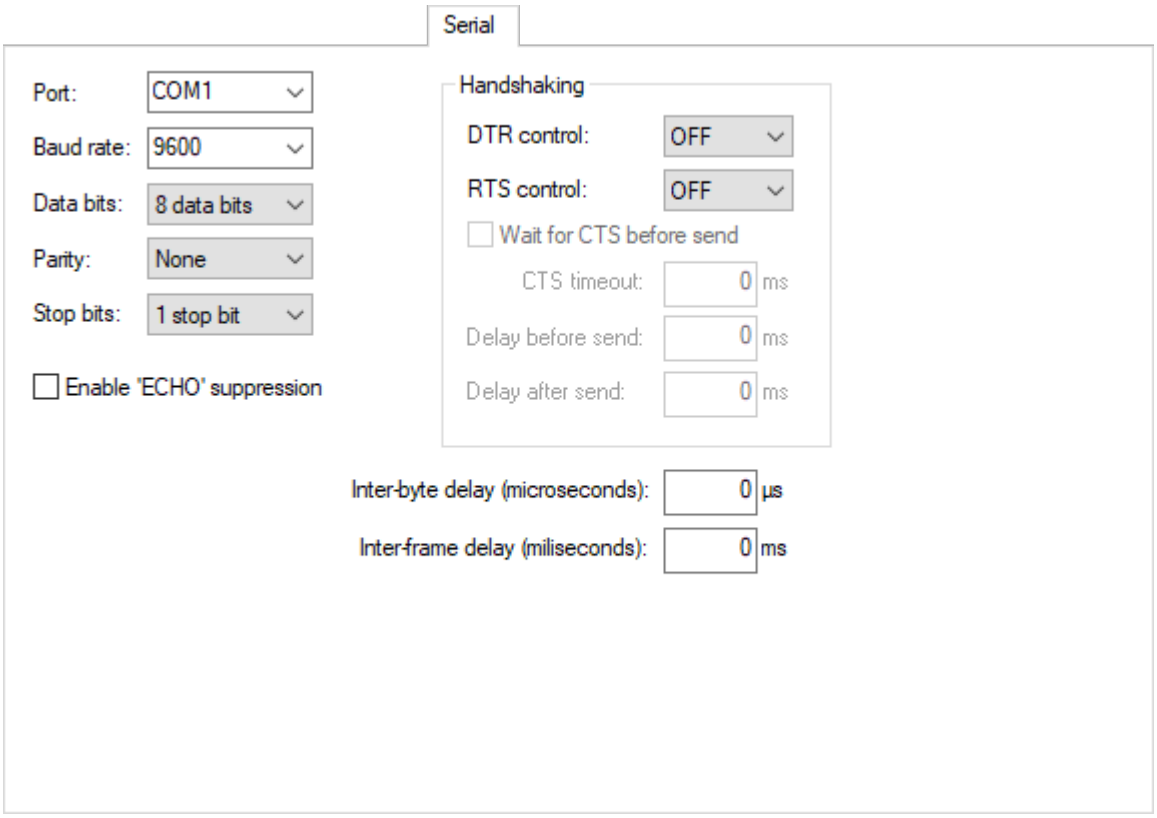
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection
Retry failed connection every ... seconds	Select this option to enable a Driver's connection retry in a certain interval, in seconds. If the **Give up after failed retries** option is not selected, this Driver keeps retrying until a connection is performed, or until the application is stopped
Give up after ... failed retries	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 **Offline** 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
Disconnect if non-responsive for ... seconds	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 **Timeout** option

Options on the Logging Options group

OPTION	DESCRIPTION
Log to File	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 **%PROCESS%** 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 **Elipse E3**, 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 **0FDAh**. Users can also use the **%DATE%** macro in the file name. In this case a log file is generated every day, in the format **aaaa_mm_dd**. 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 **%DATE_HOUR%** macro generates one log file per hour, in the format **aaaa_mm_dd_hh**
File size limit (MB)	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

Serial Tab

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



Serial tab

General options on the Serial tab

OPTION	DESCRIPTION
Port	Select a serial port on the list, from **COM1** to **COM4**, or type the name of a serial port in the format **COM*n***, 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"
Baud rate	Select a baud rate on the list (**1200**, **2400**, **4800**, **9600**, **19200**, **38400**, **57600**, or **115200**) or type a baud rate, such as 600
Data bits	Select 7 (seven) or 8 (eight) data bits on the list
Parity	Select a parity on the list. The available options are **None**, **Even**, **Odd**, **Mark**, or **List**
Stop bits	Select the number of stop bits on the list. The available options are **1**, **1.5**, or **2** stop bits
Enable 'ECHO' supression	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
Inter-byte delay (microseconds)	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
Inter-frame delay (milliseconds)	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

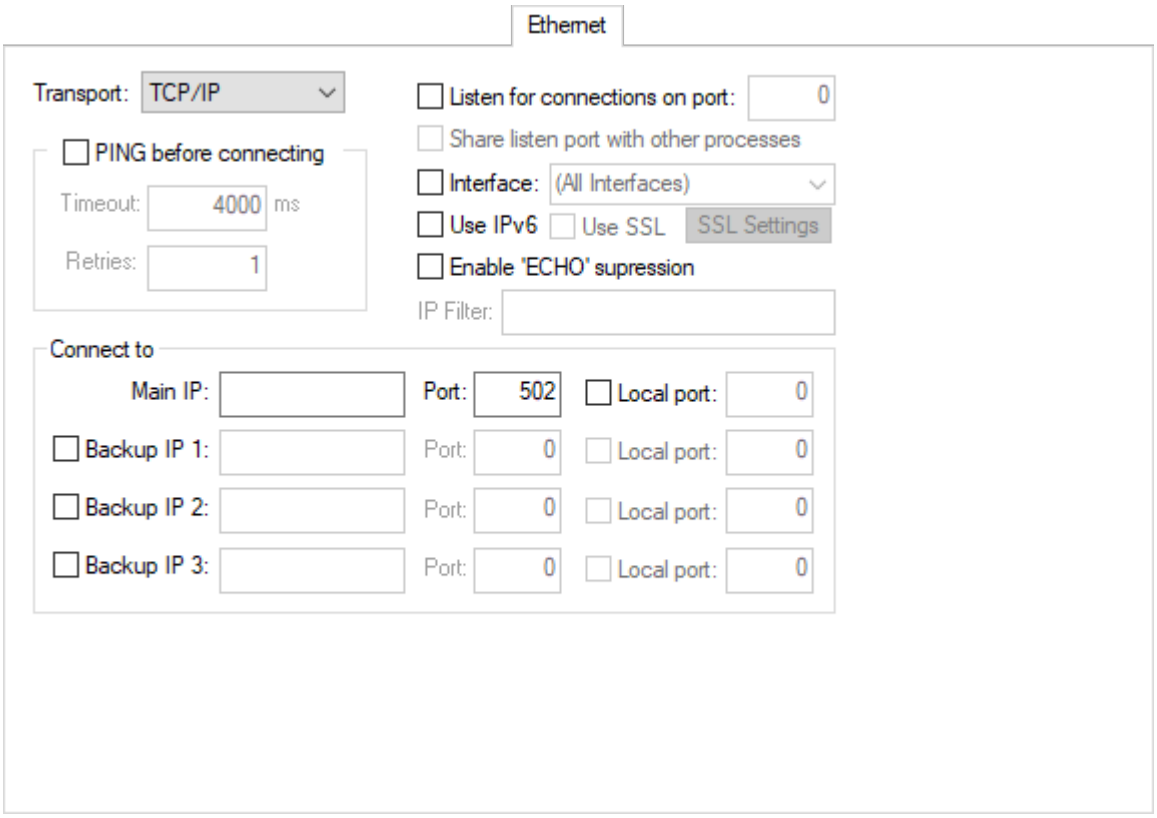
The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process, that is, it controls when data can be sent or received via serial line. Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works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

Available options on the Handshaking group

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

Available options on the Ethernet tab

OPTION	DESCRIPTION
Transport	Select the value **TCP/IP** for a TCP socket (*stream*) or select the value **UDP/IP** to use a UDP socket (*connectionless datagram*)
Listen for connections on port	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 **Connect to** option
Share listen port with other processes	Select this option to share the listening port with other Drivers and processes
Interface	Select the local network interface, identified by its IP address, that a Driver uses to establish and receive connections, or select the value **(All Interfaces)** to allow connection in any network interface
Use IPv6	Select this option to force a Driver to use addresses in **IPv6** format on all Ethernet connections. Leave this option deselected to use the **IPv4** format
Enable 'ECHO' suppression	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
IP Filter	List of restricted or allowed IP addresses from where a Driver accepts connections (*Firewall*). Please check the **IO.Ethernet.IPFilter** property for more information
PING before connecting	Enable this option to execute a **ping** 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

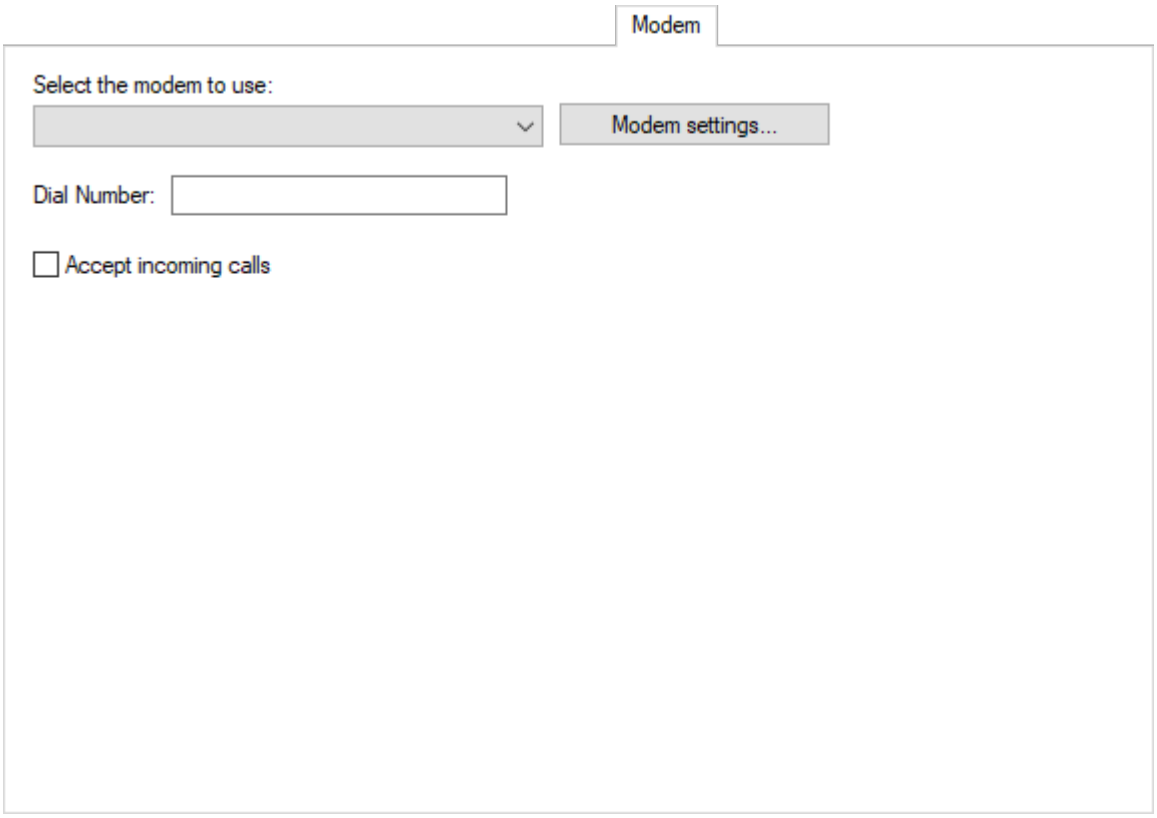
OPTION	DESCRIPTION
	of determining a successful connection before trying to open a socket with a device. The time-out of a connection with a socket can be very high. The available options are: • **Timeout**: Specify the number of milliseconds to wait for a reply from a **ping** command. Users must use a **ping** 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 • **Retries**: Number of retries of a **ping** command, not counting the first attempt. If all attempts fail, then the socket connection is aborted

Available options on the Connect to group

OPTION	DESCRIPTION
Main IP	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"
Port	Type the IP port of a remote device, between 0 (zero) and 65535
Local port	Select this option to use a fixed local IP port when connecting to a remote device
Backup IP 1, 2, and 3	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 tab

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

Available options on the Modem tab

OPTION	DESCRIPTION
Select the modem to use	Select a modem on the list of available modems on the computer. If the value **Default modem** is selected, then the first available modem is used. Selecting this option is recommended specially when an application is used on another computer
Modem settings	Click to open the configuration window of the selected modem
Dial Number	Type a default number for dialing. This value can be changed at run time. Users can use the **w** 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"
Accept incoming calls	Enable this option so that a Driver answers the phone when receiving an external call. To use this option, users must configure the **Connection management** option on the **Setup** tab to the value **Manual**

RAS Tab

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

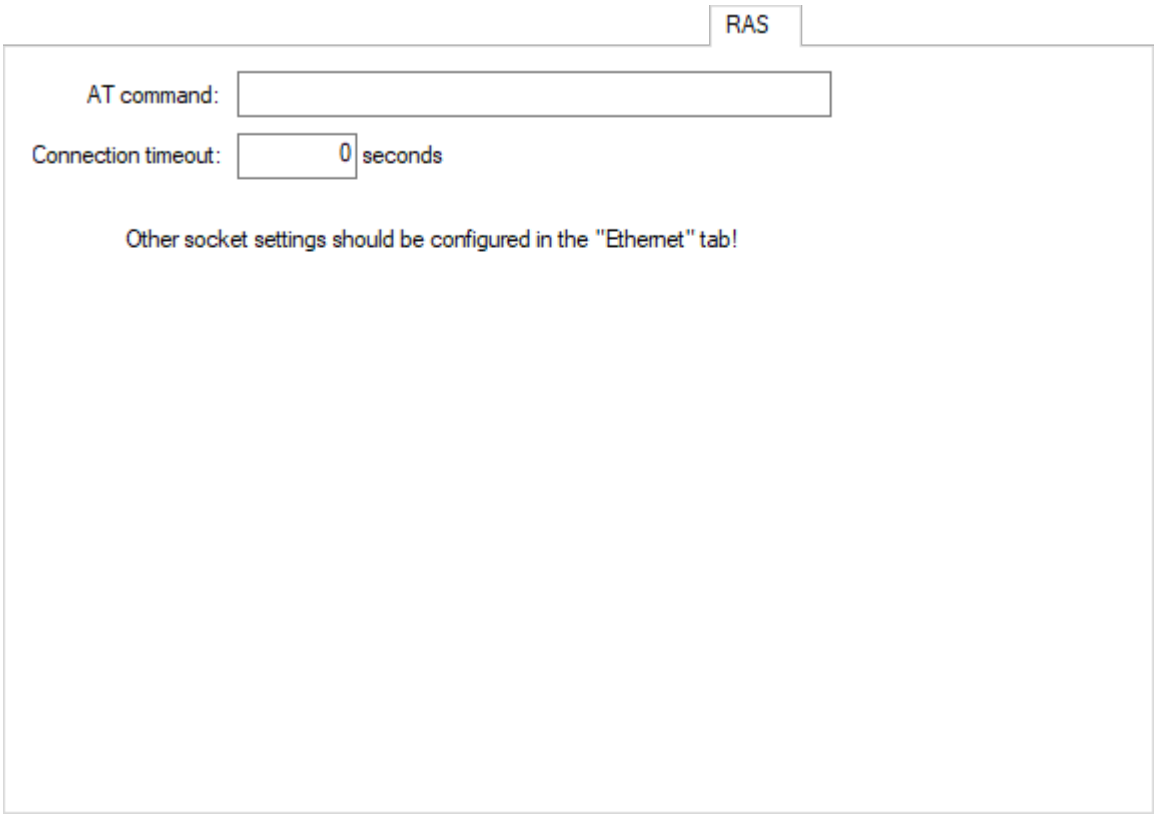
A **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

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 tab

Available options on RAS tab

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

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

Type of Tag	Block Tag
Type of Access	Read-Only
B1 Parameter	-1 (minus one)
B2 Parameter	0 (zero)
B3 Parameter	0 (zero)
B4 Parameter	1 (one)
Size Property	4 (four)
ParamItem Property	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

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	2 (two)
String Configuration	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 vlaue **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

Type of Tag	Block Tag
Type of Access	Read-Only
B1 Parameter	-1 (minus one)
B2 Parameter	0 (zero)
B3 Parameter	0 (zero)
B4 Parameter	3 (three)
Size Property	2 (two)
ParamItem Property	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

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

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

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

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

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

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

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	1101
Configuration by String	IO.Stats.Partial.BytesRecv

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

IO.Stats.Partial.BytesSent

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	1100
Configuration by String	IO.Stats.Partial.BytesSent

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

IO.Stats.Partial.TimeConnectedSeconds

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	1102
Configuration by String	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

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	1103
Configuration by String	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

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	1001
Configuration by String	IO.Stats.Total.BytesRecv

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

IO.Stats.Total.BytesSent

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	1000
Configuration by String	IO.Stats.Total.BytesSent

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

IO.Stats.Total.ConnectionCount

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	1004
Configuration by String	IO.Stats.Total.ConnectionCount

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

IO.Stats.Total.TimeConnectedSeconds

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	1002
Configuration by String	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

Type of Tag	I/O Tag
Type of Access	Write-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	4 (four)
N4 Parameter	1 (one)
String Configuration	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

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	4 (four)
N4 Parameter	2 (two)
String Configuration	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:*:*)**: 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

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

IO.Ethernet.PingTries

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

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

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	5 (five)
String Configuration	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

Type of Tag	I/O Tag
Type of Access	Write-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	1 (one)
String Configuration	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

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

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	3 (three)
String Configuration	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

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	6 (six)
String Configuration	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

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

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 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

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

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

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

IO.Serial.CTSTimeoutMs

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

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

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

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

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

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

IO.Serial.InterframeDelayMs

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

IO.Serial.Parity

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

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 COM*n* port.

IO.Serial.RTS

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

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.SupressEcho

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 Revision History

VERSION	DATE	AUTHOR	COMMENTS
3.0.2	08/22/2025	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 38034*).
3.0.1	09/30/2013	A. Quites C. Mello	• Driver ported to **IOKit** version **2.0** (*Case 14683*). • Fixed a writing problem in the **Clock Syncing: Date and Time** Tag, the *N3* parameter equal to 17, in **Broadcast** mode, the *N1* parameter equal to 900, or when writing Tags from the **D** category, in which the date was sent with the year with 4 (four) digits, while the protocol specification defines this field with 2 (two)

VERSION	DATE	AUTHOR	COMMENTS
			digits, so that devices did not recognize these commands (*Case 14666*). • Implemented a feature to send the 'XX' characters instead of a checksum on communication frames. This option can be useful when testing in HyperTerminal or simulators, when users do not need to calculate a checksum manually, or to keep compatibility with legacy applications (*Case 14684*). • Driver changed to return the value 0 (zero) for reading clock syncing Tags in **Broadcast** mode in the **T** and **D** categories (*Case 14930*). • Fixed a memory leak when closing this Driver when used in **Elipse SCADA** with the **autoscan** option enabled. This error does not occur when this Driver is used in **Elipse E3**, **Elipse Power**, or **Elipse Water** (*Case 14905*).
2.3.1	10/15/2012	C. Mello	• Added support for communication with REJ-525 relays (*Case 12967*).
2.2.1	04/27/2011	A. Quites	• Fixed a rollover error when turning the minutes of timestamps of events (*Case 12047*).
2.1.1	09/01/2010	A. Quites	• Optimized the reading of the status of Tags for Automatic Scanning of Events (*Case 11705*).
2.0.1	04/06/2010	A. Quites	• Driver ported to the **IOKit** library (*Case 11192*). • Created the automatic scanning of events using a background thread (*Case 11208*). • Added information on the user's manual about clearing events **E50** and **E51** by writing the value 0 (zero)

VERSION	DATE	AUTHOR	COMMENTS
			to the **C** category (*Case 11219*).
1.1.1	09/04/2006	C. Mello	• Adjustments to prevent internal error in this Driver during an event collecting (*Case 5141*). • General revision of all source code.
		A. Quites	• Adjustments for commands in **Broadcast** mode, that is, the *N1* or *B1* parameters equal to 900 (*Case 7329*). • Added a new Block Element to return the index of the list of **L** and **B** events (*Case 7333*).
1.0.1	12/11/2003	C. Kohlmann	• Every releases before revision control.

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

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

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

www.elipse.com.br

kb.elipse.com.br
forum.elipse.com.br
www.youtube.com/elipsesoftware
elipse@elipse.com.br

Gartner 2014 Cool Vendor

OPC FOUNDATION MEMBER

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)