Allen Bradley CIP Driver

Filename	ABCIP.dll
Manufacturer	Rockwell
Devices	Rockwell controllers from ControlLogix, FlexLogix, and CompactLogix families using an Ethernet card, Ethernet port, or Serial port
Protocol	Ethernet/IP - CIP (Control Information Protocol)
Version	3.0.23
Last Update	08/20/2020
Platform	Win32
Dependency	IOKit v1.15
Superblock Readings	No
Level	0

Introduction

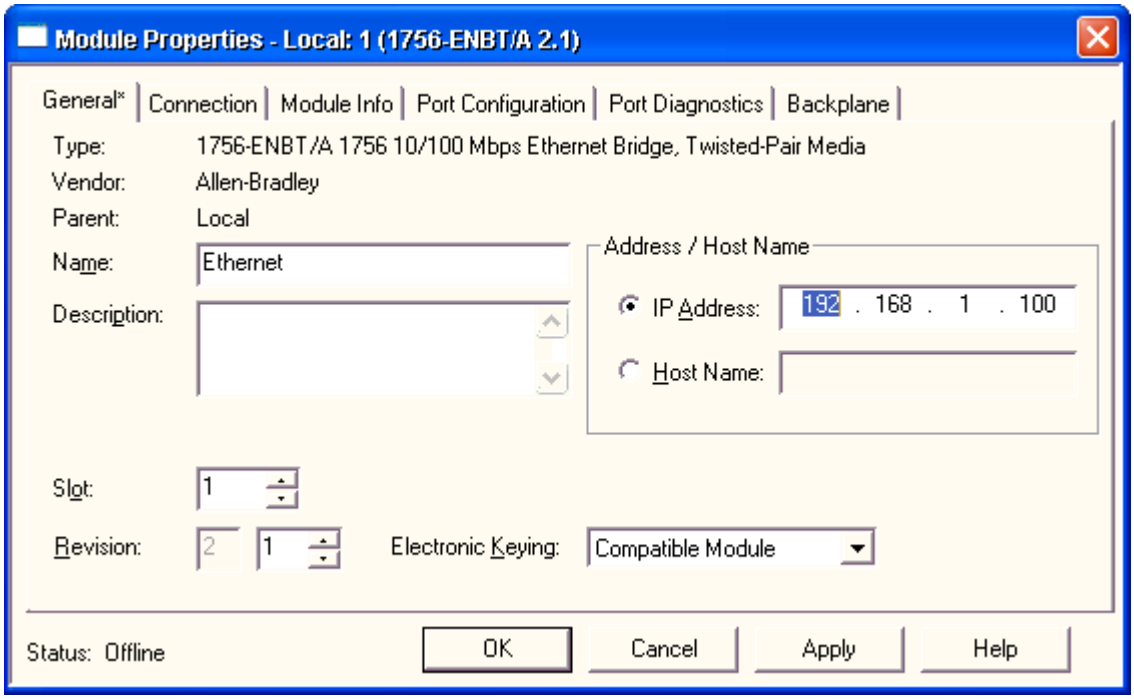
The Allen Bradley CIP Driver supports communication with Rockwell controllers from ControlLogix, FlexLogix, and CompactLogix families using an Ethernet card, Ethernet port, or Serial port.

Driver Configuration

This section provides information about the configuration of Allen Bradley CIP Driver.

Device Configuration

To communicate via Ethernet, users must inform a controller's IP address and port 44818. An IP address is defined on controller properties via RSLogix 5000 application, as shown on the next figure.



Defining an IP address on RSLogix 5000 application

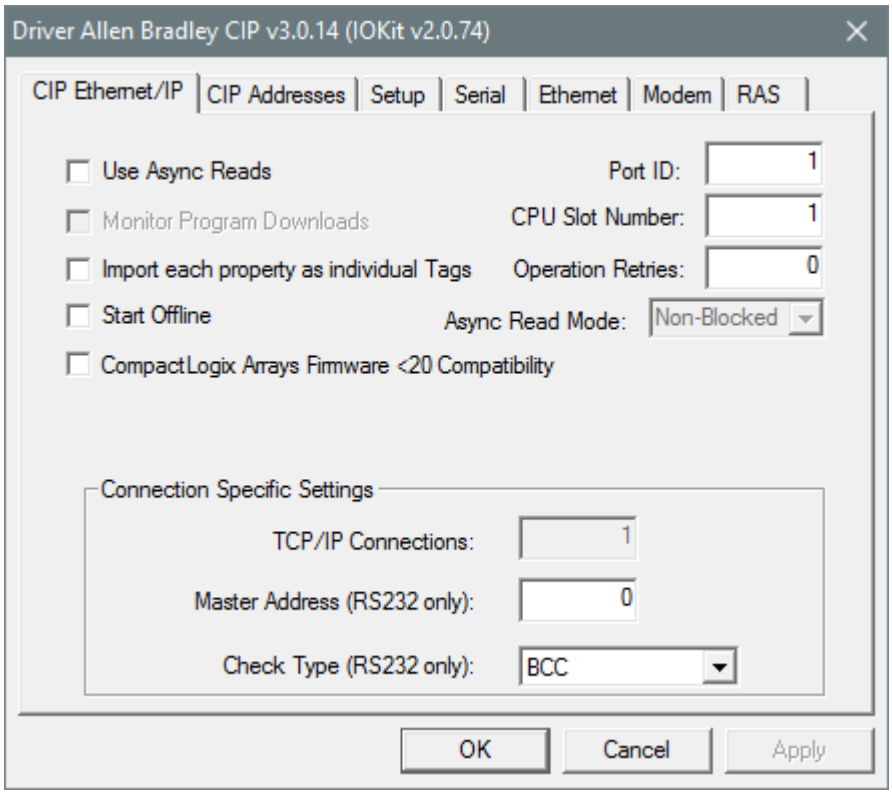
[P] Parameters Configuration

P1	Not used (leave it in zero)
P2	Not used (leave it in zero)
P3	Not used (leave it in zero)
P4	Not used (leave it in zero)

Extra Configuration

This section contains information about Allen Bradley CIP Driver's extra configurations performed on **CIP EthernetIP** and **CIP Addresses** tabs.

CIP Ethernet/IP Tab



CIP Ethernet/IP tab

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

Available options on CIP Ethernet/IP tab

OPTION	DESCRIPTION
Use Async Reads	Starting with **E3** version 3.0, operating with **IOKit** 1.15 or later, this Driver can operate with asynchronous readings, which makes readings faster
Monitor Program Downloads	When using asynchronous readings described on the previous option, this Driver requests readings from the physical address that variables occupy on PLC's memory. If during a communication there is a program download to the PLC, physical addresses may change, which may lead to incorrect readings of values. Select this option if there is a possibility of new downloads while this Driver is operating, which gets updated addresses immediately after this download finishes. **NOTE**: Using this option may turn communication approximately 2% slower
Import each property as individual Tags	When using **E3**'s Tag Browsing, users can select whether data structures, arrays, or vectors are created as Block Tags or as individual Tags. **NOTE**: If users select the option to create individual Tags, it is recommended to use asynchronous readings

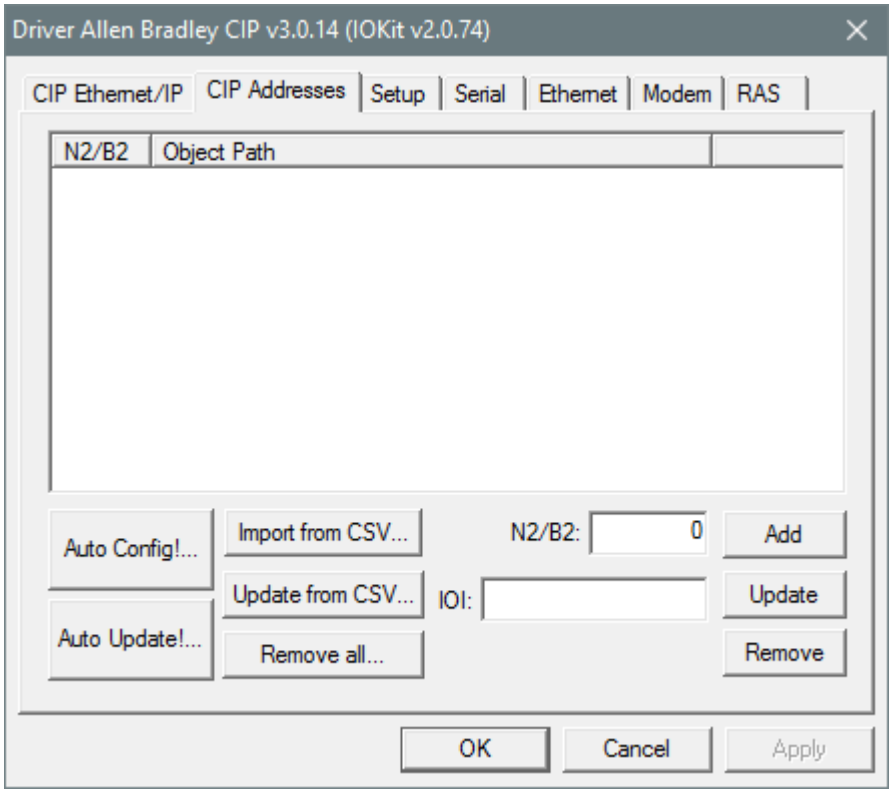
OPTION	DESCRIPTION
Start Offline	Indicates that this Driver must start without initiating the communication, so that users can change, at run time, some parameters before connection. After defining these parameters, this Driver can be freed to communicate by writing 0 (zero) to the **CIP.StartOffline** parameter, using **IOKit**'s parameter-changing services. For more information, please check topic **Changing Parameters At Run Time**
Port ID	Specifies an output from Ethernet card and it must be equal to 1 (one, port for controller's backplane)
CPU Slot Number	Slot position where the processor is, between 0 (zero) and 255
Operation Retries	Internal retries for each reading and writing operation before informing an error
Async Read Mode	When using the asynchronous readings option, informs the type of reading optimization adopted. The **Blocked** mode tries to find parents, that is, items that represent the first level on the user-defined object structure on the memory map, by reading all children. This mode is recommended when most items on each structure must be read by an application. The **Non-Blocked** mode, on the other hand, identifies each child item separately, requesting a reading on these items individually in a grouped request (usually each group contains approximately 30 items). This mode is recommended when only a few items on each user-defined object structure must be read by an application
CompactLogix Arrays Firmware <20 Compatibility	Select this option if using a CompactLogix controller with a firmware version less than or equal to 20.XX, which represents a shift on **Array** types using physical addressing (Non-Blocked)
TCP/IP Connections	Allows users to define from one to four simultaneous TCP/IP connections with a controller, aiming to improve communication performance. If communication is serial, this parameter is not used. **NOTE**: It is recommended to use asynchronous readings (please check the **Use Async Reads** option) so that connections are used more efficiently
Master Address (RS232 only)	Indicates a Driver's address (Master) for communication via serial port
Check Type (RS232 only)	Informs whether error-checking calculation when communicating through serial port is **CRC** or **BCC**

NOTE

This Driver **DOES NOT** use **E3**'s default Superblock system. This Driver's **EnableReadGrouping** option must be disabled.

CIP Addresses Tab

The **CIP Addresses** tab must be used for mapping controller's symbolic addresses to numeric *N* parameters used by **Elipse SCADA** version 2.29 or earlier.



CIP Addresses tab

For Tag addressing, users must define a link between each variable's name in ControlLogix program, called IOI (*Internal Object Identifier*), and an index to define in the *N2* parameter of each Tag. These links can be defined in three different ways:

- **Automatically by the Driver communicating with PLC**: Click **Auto Config** and provide a file name with a .csv extension where a database is stored. Each Tag found is linked to an *N2* sequential number. The **Auto Update** option performs the same operation, except that it includes only variables that do not repeat among the ones already on the list (recommended for updates performed on PLC's memory).

- **Import from a CSV file**: This file must have a column with *N2* or *B2* parameter and another column with the IOI name. Use the **Import from CSV** to create a new list from this CSV file import, or use the **Update from CSV** option to update that list by importing from a CSV file only variables that do not repeat on the existing list.

- **Directly typing on that window the N2 parameter and the corresponding IOI**: IOI must represent the **WHOLE** path of an item, including a program's name or task.

Tag Addressing Parameters

Elipse E3

- **N1**: If = 1, reading operation will happen in exclusive mode (individual polling).

  If = 2, shall be used when reading or writing strings greater than the standard ASCIISTRING82 (82 chars).

- **N2**: Not used
- **N3**: Not used
- **N4**: Not used
- **Device**: Not used
- **Item**: Variable's path on a controller

Elipse SCADA

- **N1**: Not used
- **N2**: Tag's IOI index (defined on **CIP Addresses** tab)
- **N3**: Not used
- **N4**: Not used

Supported Data Types

ControlLogix supports several internal data types, such as:

- **Boolean**, **Bit**, **Float**, **Real**, **Signed**, **Unsigned**, **Int**, etc.
- Default array types (one, two, or three dimensions)
- Logix's pre-defined structures: **Timer**, **Counter**, **Axis**, **SFC_Step**, etc.
- User-defined structures
- Arrays of structures
- Any combination of all previously described types

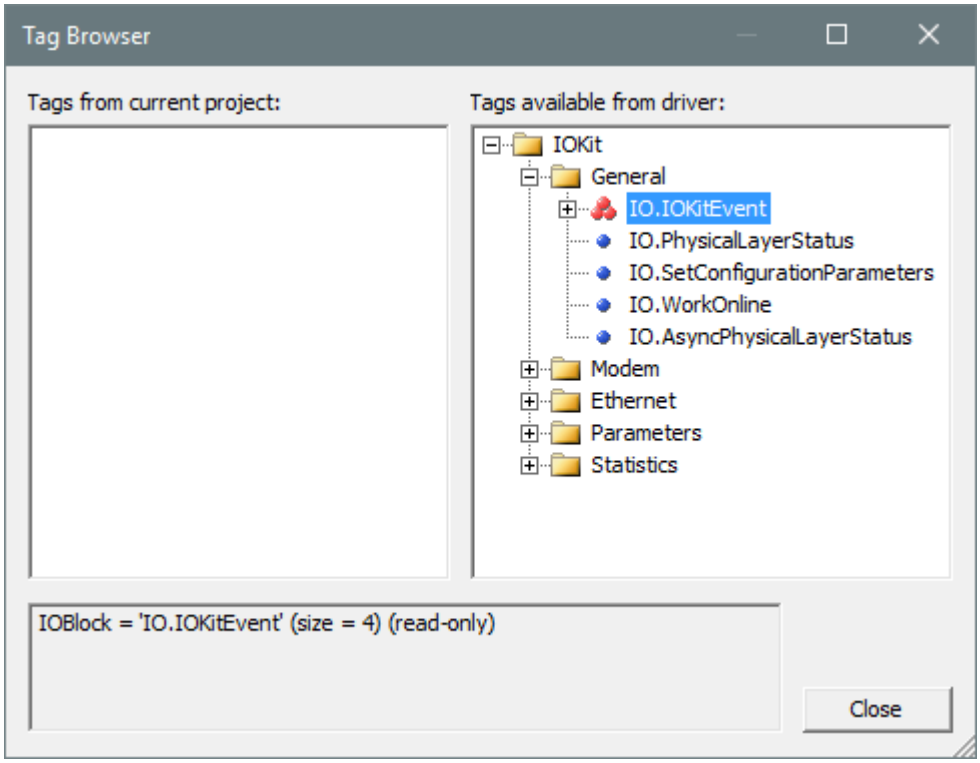
If users opted by disabling the **Import each property as individual tags** option or if they are using **Elipse SCADA**, each variable inside a structure or array is mapped to a Block Element. Variables inside Tasks are inserted into folders or Blocks with the same Task's name. Data types for each variable are defined in the controller and automatically adapted, therefore users do not need to define data types for each Tag.

If the **Import each property as individual tags** option is selected, all Tags are created in an independent way without using Blocks, following the same data structure created in the controller.

Tag Browser Window

E3 allows using an online Tag Browsing service, which allows retrieving all variables from a controller. This function is available in the **Tag Browser** option on Tag's edition window.

Before using the **Tag Browser** option, users must specify correctly a communication interface (**Ethernet** or **Serial**), as well as its parameters. To use these Tags, drag them to the **Tags from current project** list.



Tag Browser window

IOKit Tags

When used in **Ethernet** mode, default **IOKit** Tags do not work with this Driver. To perform actions and retrieve basic **IOKit** statuses, two special Tags were created, described on the next table.

Special Tags for IOKit

TAG	CONFIGURATION	DESCRIPTION
Connection Status	**Device**: Not used **Item**: IO.AsyncPhysicalLayerStatus	Returns 2 (two) if all channels (between one and four) are connected, 1 (one) if any channel is still connecting, and 0 (zero) if no channel is connected
IP Switch	**Device**: Not used **Item**: IO.Ethernet.AsyncIPSwitch	Similar to the **IOKit** Tag with the same name, when writing to this Tag the Driver switches the TCP/IP connection of all channels (between one and four) between main and backup addresses

Changing Parameters At Run Time

All parameters on the next table can be configured at run time by writing to IOKit's **Set Configuration Parameters** Tag, defined by the address -1, 0, 0, 3 (*N1.N2.N3.N4*).

Parameters at run time

PARAMETER	COMMENT	DEFAULT VALUE
CIP.StartOffline	If in 1 (one), allows this Driver to start inactive, so that other parameters are configured at run time. After defining them, this variable can be changed to 0 (zero), so this Driver starts to communicate with its new parameters	0 (zero)
CIP.CheckDownload		0 (zero)
CIP.Retries		0 (zero)
CIP.PortID		1 (one)
CIP.DefaultSlaveAddress		0 (zero)
CIP.SlotNumber		1 (one)

Example of a configuration script at run time for **E3**, assuming that this Driver was saved with the **StartOffline** option selected.

```
Dim arr(2)
//Configure Driver's name and path
Set Driver = Application.GetObject("ABCIPDriver")
arr(1) = Array("CIP.Retries", 2)
arr(2) = Array("CIP.StartOffline", 0)
Driver.Write -1, 0, 0, 3, arr
```

The following parameters cannot be changed at run time:

- **CIP.CheckDownload**
- **CIP.TCPConn**
- **CIP.UseCache**
- **CIP.TotalTargets**
- **CIP.TotalDefs**
- **CIP.ReadType**

Documentation of I/O Interfaces

This section contains the documentation of I/O Interfaces referring to the **Allen Bradley CIP Driver**.

Driver Configuration

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

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

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

1. Open the Organizer.
2. Select the Driver on Organizer's tree.
3. Click **Extras** on **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 Driver for each serial port.

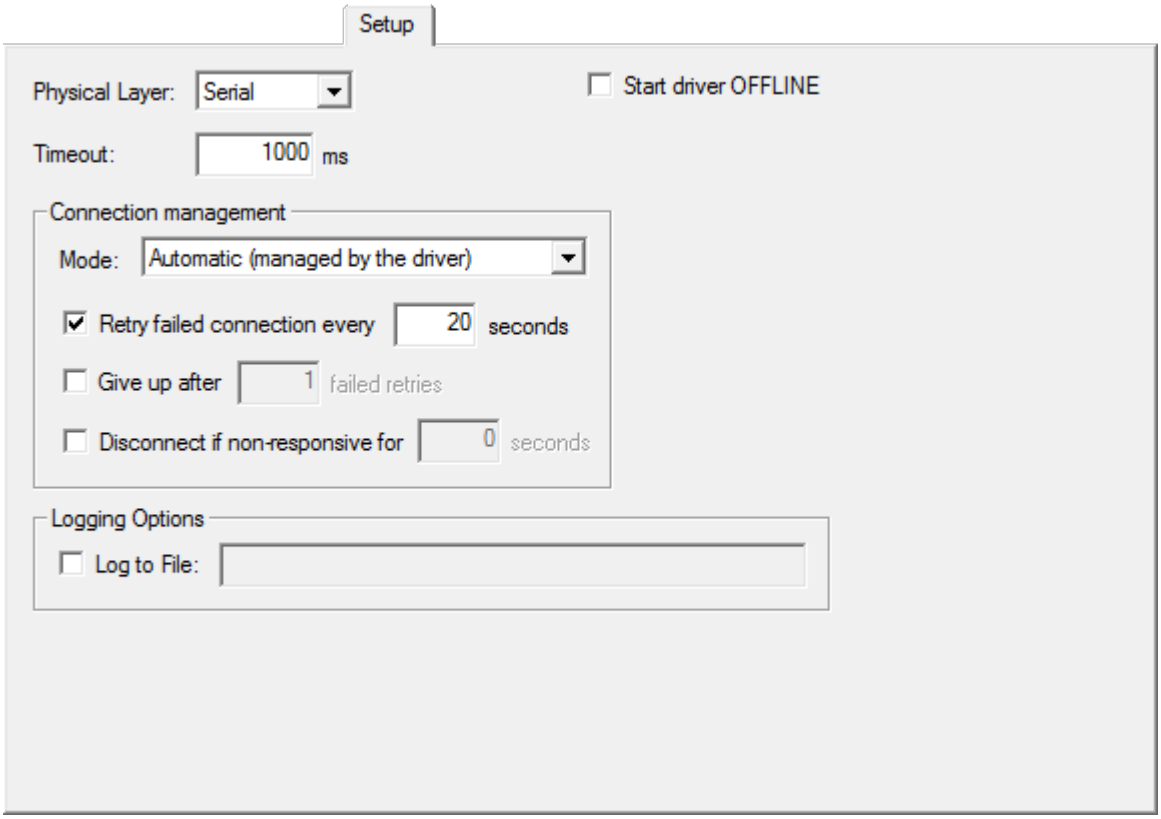
Configuration Dialog Box

The I/O Interfaces dialog box 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 each Driver) on the configuration dialog box.

Setup Tab

The **Setup** tab contains Driver's general configurations. This tab is divided into three distinct parts:

- **General configurations**: Configurations of 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 Setup tab

OPTION	DESCRIPTION
Physical Layer	Select the physical layer on the 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 a byte (any byte from reception's buffer).
Start driver OFFLINE	Select this option so that the Driver starts in **Offline** mode (stopped). This means that the I/O interface is not created until this Driver is configured to **Online** mode (using a Tag in an application). This mode enables a dynamic configuration of an I/O interface at run time. Please check topic **Working Offline** for more details.

Options on Connection management group

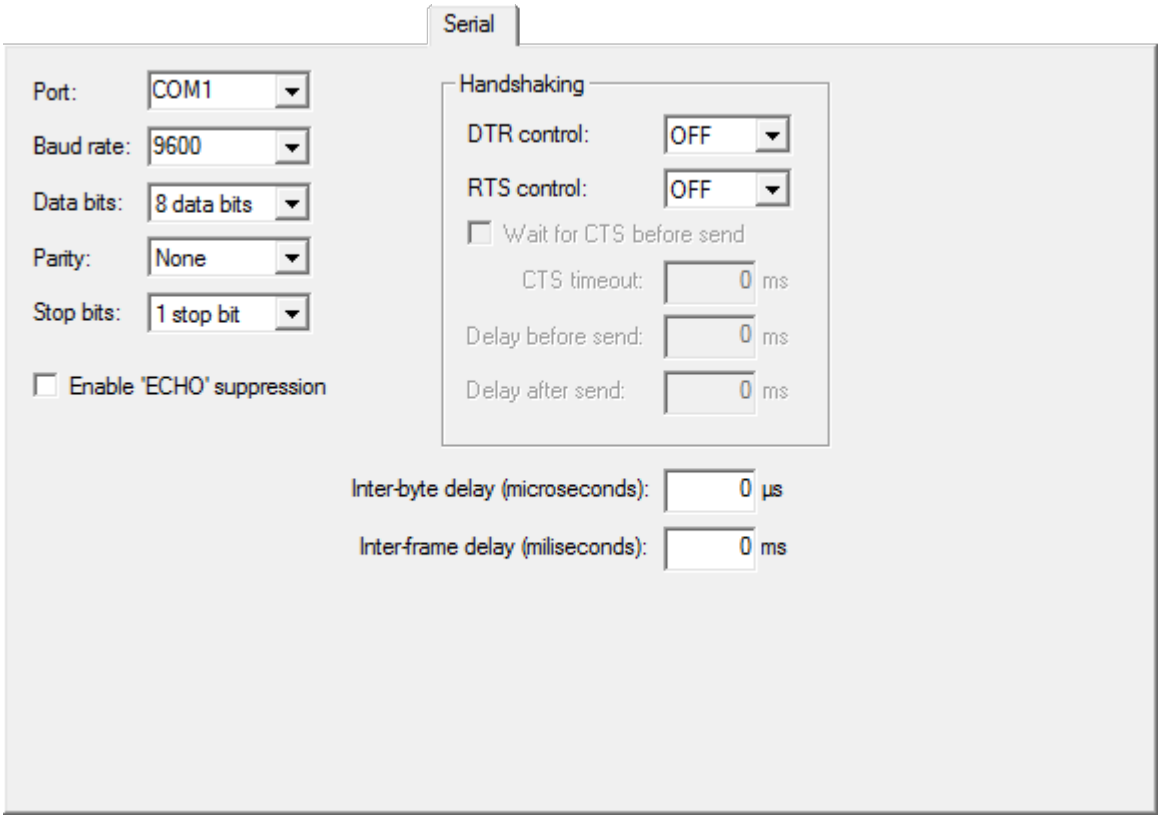
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage the connection. Please check topic **Driver Statuses** for more details.
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, the Driver keeps retrying until the 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, the 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 Logging Options group

OPTION	DESCRIPTION
Log to File	Enable this option and configure the name of the file to write the log. Log files can be large, so use this option for short periods of time, only for test and debugging purposes. If the **%PROCESS%** macro is used in the log file name, it is replaced by the ID of the current process. This option is particularly useful when using several instances of the same Driver in **E3**, thus allowing each instance to generate a separate log file. For example, when configuring this option as **c:\e3logs\drivers\sim_%PROCESS%.log**, a file **c:\e3logs\drivers\sim_00000FDA.log** is generated 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 as **c:\e3logs\drivers\sim_%DATE%.log**, a file **c:\e3logs\drivers\sim_2005_12_31.log** is generated in 12/31/2005 and a file **c:\e3logs\drivers\sim_2006_01_01.log** is generated in 01/01/2006.

Serial Tab

Use this tab to configure parameters of the **Serial** Interface.



Serial tab

General options on 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*** (for example, "COM15"). When typing a port's name manually, the dialog box only accepts port names 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 (for example, 600).
Data bits	Select 7 or 8 data bits on the list.
Parity	Select a parity on the list (**None**, **Even**, **Odd**, **Mark**, or **List**).
Stop bits	Select the number of stop bits on the list (**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 (1000000 is equal to a second). This option must be used with small delays (less than a millisecond).

OPTION	DESCRIPTION
Inter-frame delay (milliseconds)	Defines a delay between packets sent or received by the I/O Interface, in thousandths of a second (1000 is equal to a second). This delay is applied if the I/O Interface sends two consecutive packets, or between a received packet and the next sending.

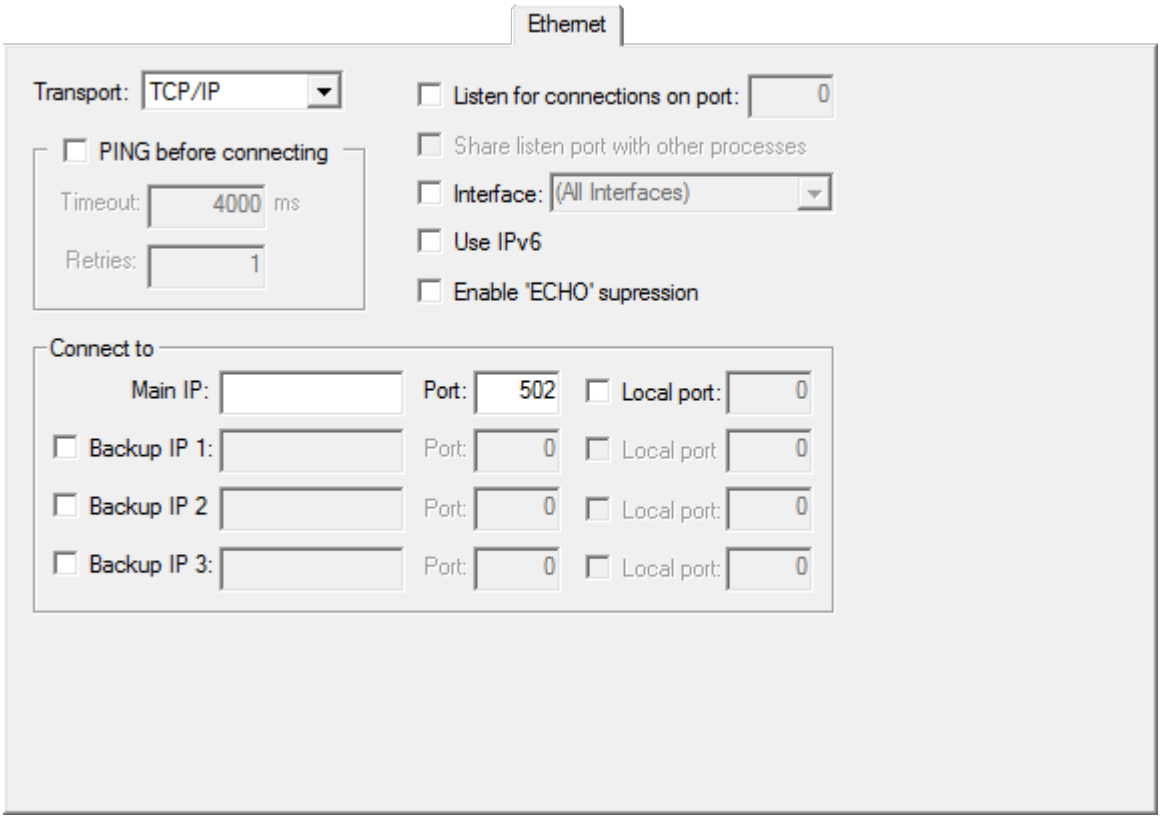
The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process (controlling when data can be sent or received via serial line). Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works with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

Available options on Handshaking group

OPTION	DESCRIPTION
DTR control	Select **ON** to keep the **DTR** signal always on while the serial port is open. Select **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 **ON** to keep the **RTS** signal always on while the serial port is open. Select **OFF** to turn the **RTS** signal off while the serial port is open. Select **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 to **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, the Driver then fails the current communication and returns an error.
Delay before send	Some serial port hardware 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 the **Ethernet** Interface. These parameters (all except port configurations) must also be configured for use in the **RAS**.



Ethernet tab

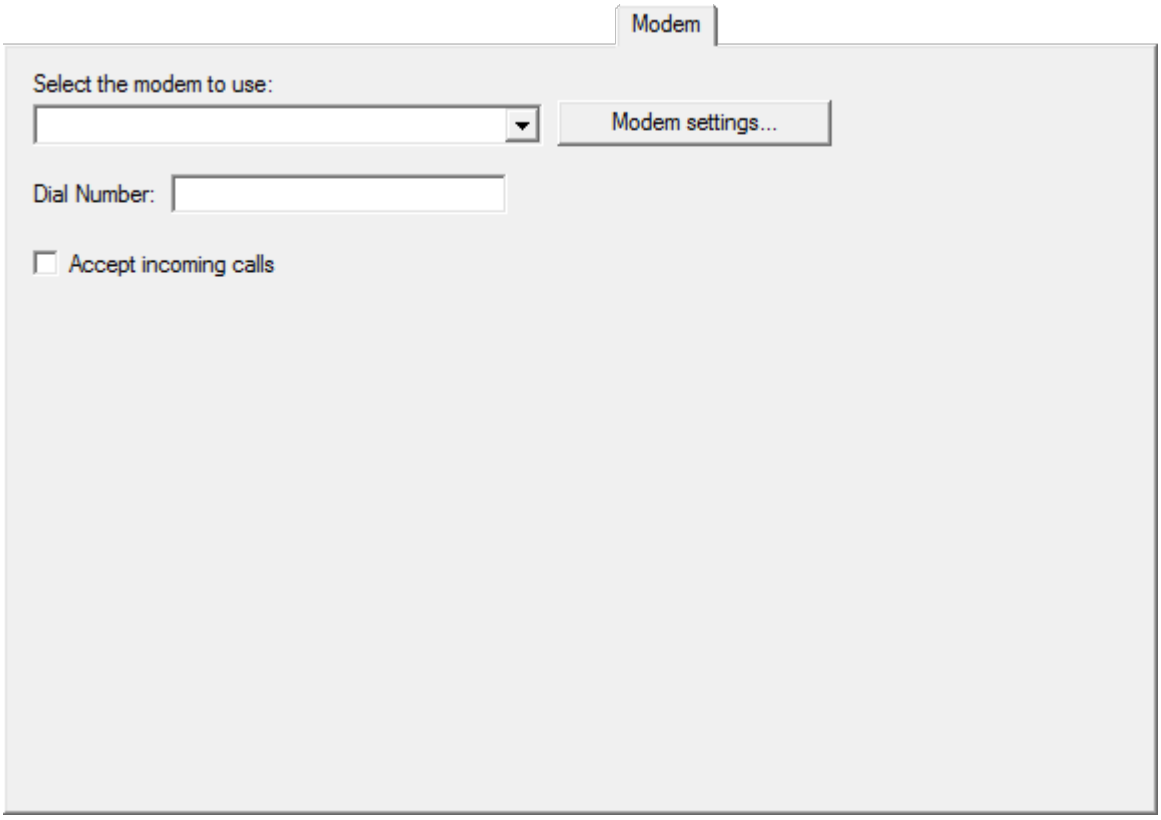
Available options on Ethernet tab

OPTION	DESCRIPTION
Transport	Select **TCP/IP** for a TCP socket (stream). Select **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, the 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 listen port with other Drivers and processes
Interface	Select the local network interface (identified by its IP address) that is used by the Driver to establish and receive connections, or select the **(All Interfaces)** item to use any local network interface
Use IPv6	Check this option to force the Driver to use IPv6 addresses on all Ethernet connections. If this option is unchecked the Driver will work with IPv4 addresses
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

OPTION	DESCRIPTION
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 details
Main IP **Backup IP 1** **Backup IP 2** **Backup IP 3**	These options allow configuring up to four IP addresses for a remote device: • **IP**: Type an IP address for the remote device. This can be an IP address separated by dots, as well as a URL. For a URL, the Driver uses the available DNS service to map that URL to an IP address. For example, "192.168.0.13" or "Server1" • **Port**: Type an IP port for a remote device (from 0 to 65535) • **Local port**: Select this option to use a fixed local port when connecting to a remote device
PING before connecting	Enable this option to execute a **ping** command (check if a device can be reached on a network) for a device before trying a socket connection. This is a quick way 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): • **Timeout**: Specify the number of milliseconds to wait for a reply from the **ping** command. Users must use the **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 (between one and 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

Modem Tab

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



Modem tab

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

Available options on Modem tab

OPTION	DESCRIPTION
Select the modem to use	Select a modem on the list of modems available on the computer. If the **Default modem** option is selected, then the first available modem is used. Selecting this option is recommended specially when the 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 (waiting for the dial tone). Por exemplo, "0w33313456" (disca o número zero, espera e então disca o número "33313456").
Accept incoming calls	Enable this option so that the Driver answers the phone when receiving an external call. To use this option, users must configure the **Connection management** option on **Setup** tab to **Manual**.

RAS Tab

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

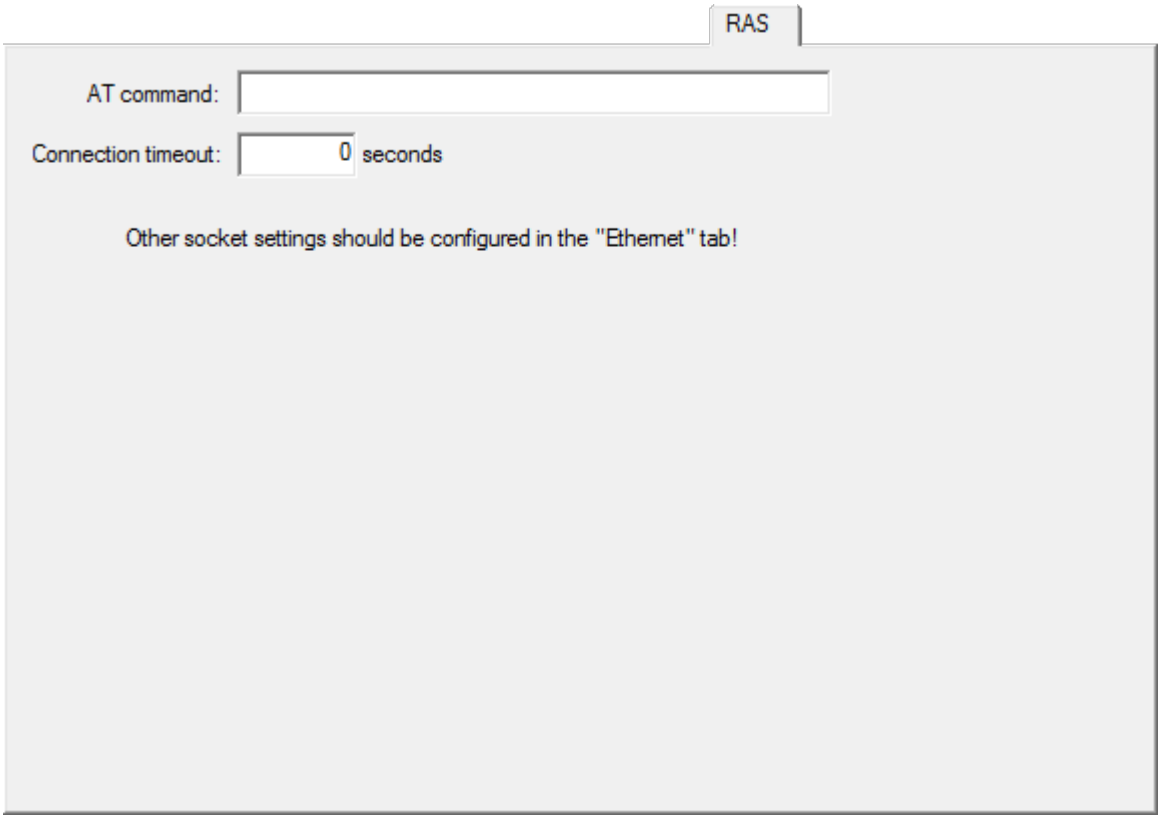
The **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

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. Clear the socket (remove any TELNET greeting message received from the RAS device).
2. Send an **AT** dial message (in ASCII) in the socket.
3. Wait 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 the device (connection was established).

If step 5 is successful, then the socket behaves as a normal socket, with the RAS device working as a router between the Driver and the device. Bytes sent by the 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 the Driver using the same socket.

After establishing the connection, the **RAS** interface monitors data received by the Driver. If a **String** "NO CARRIER" 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 **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" (tone dialing 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.IOKitEvent

Type of Tag	Block Tag
Type of Access	Read-Only
B1 Parameter	-1
B2 Parameter	0
B3 Parameter	0
B4 Parameter	1
Size Property	4
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
  - **0**: Information
  - **1**: Warning
  - **2**: Error
- **Element 1**: Source of event
  - **0**: Driver (specific of a Driver)
  - **-1**: IOKit (generic events of I/O Interfaces)
  - **-2**: **Serial** Interface
  - **-3**: **Modem** Interface
  - **-4**: **Ethernet** Interface

- **-5**: **RAS** Interface

- **Element 2**: Error number (specific for each source of event)

- **Element 3**: Event message (**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
N2 Parameter	0
N3 Parameter	0
N4 Parameter	2
String Configuration	IO.PhysicalLayerStatus

This Tag indicates the status of the physical layer. Its possible values are the following:

- **0**: Physical layer stopped (the 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 (the Driver is in **Online** mode, but the physical layer is not connected. If the **Connection management** option is configured as **Automatic**, the physical layer can be connecting, disconnecting, or waiting for a reconnection attempt. If the **Connection management** option is configured as **Manual**, then the physical layer remains in this status until forced to connect)

- **2**: Physical layer connected (the physical layer is ready for use). This **DOES NOT** mean the device is connected, only the access layer is working

IO.SetConfigurationParameters

Type of Tag	Block Tag
Type of Access	Read-Only
B1 Parameter	-1
B2 Parameter	0
B3 Parameter	0
B4 Parameter	3
Size Property	2
ParamItem Property	IO.SetConfigurationParameters

Use this Tag to change any property of Driver's configuration dialog box at run time (the complete list of properties can be found on the specific topic of each Interface).

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 Driver's configuration dialog box. Users can write to a PLC Tag or to a Block Tag containing the parameters to change (writings of individual Block Elements are 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 three parameters, then the size of the Block must be 6 (3 * 2). The first Element is the property's name (as a **String**) and the second Element is the property's value. Check this script in **Elipse SCADA**:

```
// 'Block' must be a Block Tag with automatic reading,
// scan reading, and automatic writing 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 **E3**, the ability to create arrays at run time allows using an I/O Tag as well as a Block Tag. Users can use Driver's **Write** method to send all parameters to the Driver, without creating a Tag. Check these examples:

```
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 elements of the array are ignored by the 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 Driver's log or use the *writeStatus* parameter of the **WriteEx** method to find out the exact cause of the 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 "Failure when configuring Driver parameters: " + strError
End If
```

IO.WorkOnline

Type of Tag	I/O Tag
Type of Access	Reading or Writing
N1 Parameter	-1
N2 Parameter	0
N3 Parameter	0
N4 Parameter	4
String Configuration	IO.WorkOnline

This Tag informs Driver's current status and allows starting or stopping the physical layer.

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

In the next example (using **E3**), the Driver is configured to **Offline** mode, its COM port is changed, and then configured to **Online** mode again:

```
' Configure to Driver 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 Driver to Online mode
Driver.Write -1, 0, 0, 4, 1
```

The **Write** method can fail when configuring the Driver to **Online** mode (writing the value one). In this case, the 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)
- Driver may have run out of memory
- Physical layer probably did not create its working thread (search the log file for the message "Failed to create physical layer thread!")
- Physical layer could not start. The cause of failure depends on the type of physical layer. It can be an invalid serial port number, failure when starting Windows Sockets, failure when starting TAPI (modem), etc. 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 (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 the Connection:

- **0**: Automatic mode (the Driver manages the connection)
- **1**: Manual mode (the application manages the connection)

IO.GiveUpEnable

When configured to True, defines a maximum number of reconnection attempts. If all reconnection attempts fail, the Driver enters the **Offline** mode. When configured to False, the 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), the Driver tries only one reconnection when the reconnection is lost. If this one fails, the 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 inactivity. If the physical layer is inactive for this period of time, it is 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 (the 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. The 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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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 **Ethernet** Interface.

I/O Tags

Tags of Ethernet Interface (N2/B2 = 4)

The Tags described next allow controlling and identifying the **Ethernet** Interface at run time (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
N4 Parameter	0 (zero)
String Configuration	IO.Ethernet.IPSelect

Indicates the active IP address. Possible values are the following:

- **0**: The main IP address is selected

- **1**: The alternative (backup) IP address is selected
- **2**: The alternative (backup) IP address 2 is selected
- **3**: The alternative (backup) IP address 3 is selected

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

During the connection process, if the active IP address is not available, the I/O Interface tries to connect using the next alternative 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 IP address or **1, 2, 3**: Alternative IP address) if the Driver is currently connected. If the Driver is disconnected, this Tag configures the active IP address for the next connection attempt.

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
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 Driver tries to connect to each one of the alternative (backup) IP addresses and back to the main IP address until a connection is established.

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

Properties

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

NOTE

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

IO.Ethernet.AcceptConnection

◩ Configure to False if the Driver must not accept external connections (the Driver behaves as a master) or configure to True to enable the reception of connections (the Driver behaves as a slave).

IO.Ethernet.BackupEnable

◩ Configure to True to enable the alternative (backup) IP address. If the reconnection attempt with the main IP address fails, the Driver tries to use the alternative IP address. Configure to False to disable its usage.

IO.Ethernet.BackupLocalPort

Local port number to be used when connecting to the alternative IP address of the destination device. Used only if **IO.Ethernet.BackupLocalPortEnable** is True.

IO.Ethernet.BackupLocalPortEnable

Configure to True to force the use of a specific local port when connecting to the backup IP address of a remote device. Configure to False to let windows find dinamically an available local port.

IO.Ethernet.BackupIP

Alternative (backup) IP address of the destination 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.BackupPort

Port number of the alternative IP address of the destination 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 the 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 the IP addresses. To block an IP address or a range of IP addresses, use the tilde ("~") character at the beginning of the address. 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 range from 192.168.0.41 to 192.168.0.50
- **192.168.0.***: Accepts connections from IPv4 addresses in the range from 192.168.0.0 to 192.168.0.255
- **fe80:3bf:877::*:* (expands to fe80:03bf:0877:0000:0000:0000:*:*)**: Accepts connections from IPv6 addresses in the range from fe80:03bf:0877:0000:0000:0000:0000:0000 to 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 range from 192.168.0.0 to 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 the 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 the message "Blocked incoming socket connection from {IP}!".

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

IO.Ethernet.ListenIP

IP address of the local network interface that the Driver uses to establish and receive connections. Leave this property empty to use any local network interface.

IO.Ethernet.ListenPort

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

IO.Ethernet.MainIP

IP address of the destination 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 the destination 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. Configure to False to use any available local port.

IO.Ethernet.MainPort

9 Number of the IP port on the destination 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 the destination 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 the listen port with other Drivers and processes or False to open the listen port in exclusive mode. To successfully share a listen port, all Drivers and processes that use that port must open it in shared mode. When a listen 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 in 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 the following:

- **T or TCP**: Uses the TCP/IP protocol
- **U or UDP**: Uses the UDP/IP protocol

IO.Ethernet.UseIPv6

◩ Configure in True to use IPv6 addresses on all Ethernet connections. Configure in False to use IPv4 addresses (this is the default value).

Modem Interface Configuration

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

I/O Tags

Tags of Modem Interface (N2/B2 = 3)

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

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

IO.TAPI.ConnectionBaudRate

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1
N2 Parameter	0
N3 Parameter	3
N4 Parameter	5
String Configuration	IO.TAPI.ConnectionBaudRate

Indicates a baud rate value for the current connection. If the 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
N2 Parameter	0
N3 Parameter	3
N4 Parameter	1
String Configuration	IO.TAPI.Dial

Write any value to this Tag to force the **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
N2 Parameter	0
N3 Parameter	3
N4 Parameter	4
String Configuration	IO.TAPI.HangUp

Any value written to this Tag turns the current call off.

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, the Driver immediately tries to reestablish the connection.

IO.TAPI.IsModemConnected

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1
N2 Parameter	0
N3 Parameter	3
N4 Parameter	3
String Configuration	IO.TAPI.IsModemConnected

This Tag indicates modem's connection status. Possible values are the following:

- **0**: The modem is not connected, but it may be performing or receiving an external call
- **1**: The modem is connected and the 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
N2 Parameter	0
N3 Parameter	3
N4 Parameter	6
String Configuration	IO.TAPI.IsModemConnecting

This Tag indicates the connection status of a modem, with more details than the **IO.TAPI.IsModemConnected** Tag. Possible values are the following:

- **0**: Modem is not connected
- **1**: Modem is connecting (performing or receiving an external call)
- **2**: Modem is connected. While in this status, the physical layer can send or receive data
- **3**: Modem is disconnecting the current call

IO.TAPI.ModemStatus

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1
N2 Parameter	0
N3 Parameter	3
N4 Parameter	2
String Configuration	IO.TAPI.ModemStatus

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

- **"No status!"**: **Modem** Interface was not open yet or was already closed
- **"Modem initialized OK!"**: **Modem** Interface was initialized successfully
- **"Modem error at initialization!"**: Driver could not initialize modem's line. Check Driver's log file for more details
- **"Modem error at dial!"**: Driver could not start or accept a call
- **"Connecting..."**: 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!"**: Driver connected successfully (completed or accepted an external call)
- **"Disconnecting..."**: Driver is turning the current call off

- **"Disconnected OK!"**: Driver turned the current call off
- **"Error: no dial tone!"**: Driver aborted a call because the available line signal was not detected
- **"Error: busy!"**: Driver aborted a call because the line was busy
- **"Error: no answer!"**: 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
N2 Parameter	0
N3 Parameter	3
N4 Parameter	0
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 **Modem** (TAPI) Interface.

IO.TAPI.AcceptIncoming

9 Configure to False if the modem cannot accept external calls (the Driver behaves as a master) and configure to True to enable receiving calls (the Driver behaves as a slave).

IO.TAPI.ModemID

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

NOTE

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

IO.TAPI.PhoneNumber

A The telephone number used by **Dial** commands. For example, "0w01234566" (the "w" character forces the modem to wait for a call signal).

RAS Interface Configuration

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

I/O Tags

Tags of RAS Interface (N2/B2 = 5)

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

Properties

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

NOTE
The **RAS** Interface uses the **Ethernet** Interface, which for this reason must be also configured.

IO.RAS.ATCommand

AT command to send through a socket to force a RAS device to perform a call using the current RAS channel. Example: "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 **Serial** Interface.

I/O Tags

Tags of Serial Interface (N2/B2 = 2)

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

Properties

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

IO.Serial.Baudrate

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

IO.Serial.CTSTimeoutMs

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

IO.Serial.DataBits

Specifies the number of data bits to configure the serial port. Possible values are the following:

- **5**: Five data bits
- **6**: Six data bits
- **7**: Seven data bits
- **8**: Eight data bits

IO.Serial.DelayAfterMs

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

IO.Serial.DTR

Indicates how a Driver deals with the **DTR** signal:

- **OFF**: **DTR** signal is always turned off
- **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 the **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 the serial port. Possible values are the following:

- **E or Even**: Even parity
- **N or None**: No parity
- **O or Odd**: Odd parity
- **M or Mark**: Mark parity
- **S or Space**: Space parity

IO.Serial.Port

Number of the local serial port:

- **1**: Uses the COM1 port
- **2**: Uses the COM2 port
- **3**: Uses the COM3 port
- **n**: Uses the COM*n* port

IO.Serial.RTS

Indicates how a Driver deals with the **RTS** signal:

- **OFF**: **RTS** signal always off
- **ON**: **RTS** signal always on
- **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 the serial port. Possible values are the following:

- **1**: One stop bit
- **2**: One and a half stop bit
- **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 to **Toggle**.

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
3.0.23	08/20/2020	C. Mello	• Reading and Writing long strings (Case 29327)
3.0.21	09/11/2019	M. Salvador	• Fixed write problem caused on previous version (Case 27554) • Variables doesn't update in specific circumstances (Case 27549)
3.0.20	08/19/2019	C. Mello	• Driver source code platform update (*Case 27381*).

VERSION	DATE	AUTHOR	COMMENTS
3.0.19	09/17/2018	M. Salvador	• Adjustments in the connection process with the controllers (*Case 25138*).
3.0.18	09/05/2018	M. Salvador	• Added cache system for controller settings (*Case 24866*). • Improvements in the network reconnection process (*Case 24790*).
3.0.14	12/06/2016	M. Salvador	• Fixed the reading of 64-bit integer data types (*Case 20933*).
3.0.12	02/26/2016	M. Salvador	• Now if one or more services from a **Non-Blocked** request returns an error, only Tags corresponding to those services return a communication error (*Case 20505*).
3.0.11	01/28/2015	M. Salvador	• Created a mechanism to update the IP address at run time, **StartOffline** (*Case 14612*). • Removed the sending of controller's identification message at each connection (*Case 17847*).

VERSION	DATE	AUTHOR	COMMENTS
3.0.7	09/08/2014	M. Salvador	• Improvements in the process of reconnection with a controller after a failure (*Case 17088*). • Fixed the behavior of the option to monitor program download (*Case 16353*). • Fixed a problem of increasing memory usage when performing many writings continuously (*Case 16369*). • Implemented a greater tolerance to communication failures during the initialization process (*Case 17076*). • Created a new configuration option allowing to add a four-byte offset in array communication in controller's firmware versions less than or equal to 20, in the **Non-Blocked** option (*Case 17082*).
3.0.4	01/28/2014	M. Salvador	• Now each TCP/IP channel contains a KeepAlive control that keeps the connection active by using an exchange of identification messages at every 10 seconds (*Case 15623*). • Driver ported to **IOKit** v2.00 (*Case 14026*).
2.2.1	02/22/2013	M. Salvador	• If there is a communication failure during the initialization process, the Driver now discards all data structures partially obtained and restarts the initialization process (*Case 13530*).

VERSION	DATE	AUTHOR	COMMENTS
2.1.1	04/18/2011	M. Salvador M. Ludwig	• Fixed a possible wrong bit value (*Case 11951*). • Implemented multiple TCP/IP connections (*Case 11109*). • Implemented a verification of program download. • Added support for user-defined **String** data types and also support for using **Strings** in arrays or structs. • Implemented asynchronous readings (*Case 11109*). • Fixed a problem with certain data structures not being correctly mapped by this Driver (*Case 10109*). • Implemented the **Auto Update** option (*Case 10026*). • Improvements on **CIP Addresses** interface (*Case 10026*). • Created special **IOKit** Tags for when this Driver uses TCP/IP communication (*Case 11387*).
1.6.1	10/10/2006	M. Salvador	• Tag Browser can now be opened regardless the configuration window is already open. • Fixed reading errors for arrays of structures. • Tags read by Tag Browser now are displayed in alphabetical order. • Added a progress bar on Tag Browser window to indicate the progress of identifying Tags.
1.5.1	09/25/2006	M. Salvador	• Source code revision.
1.0.1		M. Salvador	• Driver's initial version.

Headquarters
Rua 24 de Outubro, 353 - 10º andar
90510-002 Porto Alegre
Phone: (+55 51) 3346-4699
Fax: (+55 51) 3222-6226
E-mail: elipse-rs@elipse.com.br

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)