Allen-Bradley ABDF1 Driver

File Name	ABDF1.dll
Manufacturer	Allen-Bradley (Rockwell Automation)
Devices	PLC5, SLC500, MicroLogix, and ControlLogix
Protocol	DF1, DF1 over CSPv4, PCCC, and Ethernet/IP
Version	2.0.18
Last Update	08/22/2025
Platform	Win32
Dependencies	IOKit version 2.0 or later
Superblock Readings	Yes
Level	0

Introduction

The Allen-Bradley ABDF1 Driver allows serial communication between **Elipse Software** systems and programmable controllers by Allen-Bradley (Rockwell Automation) compatible with DF1 protocol (Serial) and via Ethernet, directly via Ethernet channel for the SLC5/05 model and via 1761-NET-ENI module for PLC5, MicroLogix, and SLC5/0X models.

For ControlLogix devices, users must preferably use the *Allen-Bradley Ethernet/IP (ABCIP) Driver*, but users can also use this Driver by mapping ControlLogix device's internal variables to the **DF1** format.

Preparing a Device

This section contains information about the configuration of this Driver's **[P]** parameters.

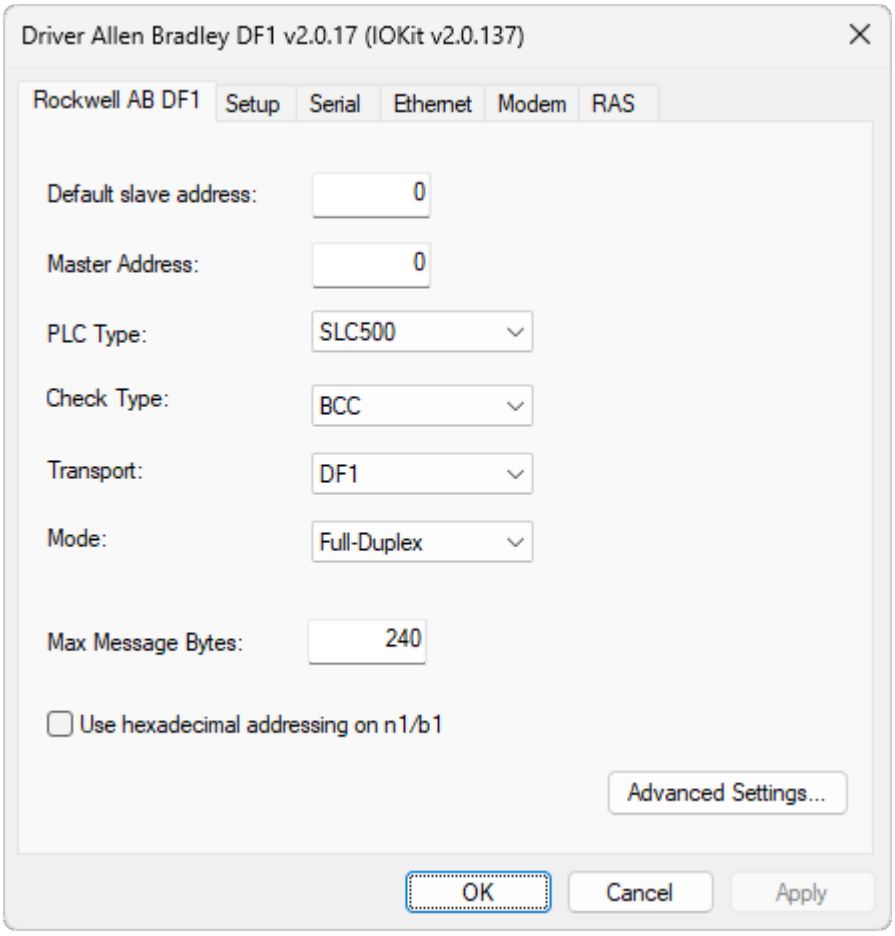
[P] Parameters for Driver Configuration

P1	Not used
P2	Not used
P3	Not used
P4	Not used

The Allen-Bradley ABDF1 Driver uses the **IOKit** library, which allows communicating via RS232 (DF1), Ethernet (DF1), Ethernet CSP, Ethernet/IP, and Ethernet/IP with PCCC messages. All parameters are defined on **IOKit**'s configuration window. For more information about the **IOKit** library, please check topic **Documentation of I/O Interfaces**.

Driver's Configuration Window

By using this Driver's configuration window, shown on the next figure, users can inform general configurations.



Rockwell AB DF1 tab

On the **Rockwell AB DF1** tab there are specific configurations for this Driver. For more information about the other tabs, please check topic **Documentation of I/O Interfaces**. The next table describe all configuration options for this tab.

Available options on the Rockwell AB DF1 tab

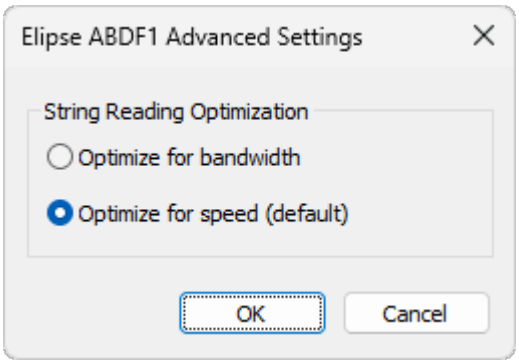
OPTION	DESCRIPTION
Default slave address	Default address of a PLC. Used whenever the addressing of Tags does not inform an address. For **Ethernet CSP** and **Ethernet/IP (ENIP)** communication, this address is always 0 (zero)
Master Address	Default address of a **Master**. For **Ethernet** communication, this address is 0 (zero)
PLC Type	Type of a PLC. The available options are **SLC500**, **PLC5**, **MicroLogix 1100** or **MicroLogix 1500** (local Ethernet port), and **MicroLogix** or **ControlLogix** (via Ethernet converter, emulating DF1)
Check Type	Type of verification. The available options are **BCC** or **CRC** (only for DF1)
Transport	Type of transport. The available options are **DF1** (usually for serial communication), **CSPv4** (Ethernet) for SLC5/05, and **ENIP** if connected via 1761 NET-ENI converter or via Ethernet port of Micrologix 1100
Mode	Inform whether the **Half-duplex** or the **Full-duplex** mode is used. This option is only relevant if the **DF1** format is used as a transport layer

OPTION	DESCRIPTION
Max Message Bytes	This parameter defines the maximum number of bytes of the data area for each communication frame. In case of blocks with more bytes, this Driver divides communication into more frames, each one containing the limit defined in this option. The default value is 240. Check a device's documentation if this value can be increased. If in doubt, use the default value
Use hexadecimal addressing on n1/b1	If this option is enabled, the *N1* and *B1* parameters of Tags are changed using an addressing in hexadecimal format. The most significant byte (MSB) indicates a destination PLC, while the least significant byte (LSB) indicates a file number. With this option enabled, users can address up to 256 PLCs, which was not possible in previous versions, as the maximum value of *N* and *B* parameters is 65535 (16 bits). With this option enabled, users must provide the value of *N1* or *B1* parameters always in hexadecimal format, adding an "h" character in **Elipse SCADA** or the prefix "&H" in **Elipse E3**, in **Elipse Power**, or in **Elipse Water**. For more information, please check *VBScript Language Reference*. Leave this option disabled if users want to keep compatibility with previous version of this Driver
Advanced Settings	Opens this Driver's **Advanced Settings Window**, described on the next topic

The remaining parameters, as already mentioned, must be informed on **IOKit** library tabs. For serial communication, the default is **19200 bps**, **8 databits**, **1 stopbit**, **no parity**, and **check type** equal to **CRC**.

Advanced Settings Window

This Driver's advanced settings window, shown on the next figure, is opened by clicking **Advanced Settings** on **Rockwell AB DF1** tab on this Driver's **Configuration Window**.



Elipse ABDF1 Advanced Settings window

This window gathers rarely used settings, which usually do not require user's attention and can be kept with their default options. Currently, the following advanced settings are available:

- **String Reading Optimization**
  - **Optimize for bandwidth**: For every **String** reading, this Driver performs 2 (two) requests to a device. The first request only reads a number value that indicates the real size of that **String**, and the second request reads that **String** with this detected size. This procedure tends to save bandwidth, because it reads the exact size of a **String**. This was the default behavior of this Driver up to version **2.0.3**

- **Optimize for speed (default)**: This is the default option for new instances of this Driver, created with libraries from version **2.0.3** or later. In this option, this Driver executes a single request for each **String** reading, always reading the maximum allowed size, 82 characters, and later discarding unused bytes. This method tends to double the speed of reading **Strings** comparing to the previous option, but it may require reading more characters in this single reading, always 82 characters, than needed

NOTE

When updating this Driver's library in instances initially created with versions earlier than **2.0.3** in older applications, this Driver keeps its legacy behavior, with the **Optimize for bandwidth** option selected.

Tag Reference

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

[N] Parameters for Addressing PLC-Type Tags

N1	Address of a PLC (PC source) × 1000 + the number of a file. If a hexadecimal addressing is used, **MSB** is equal to the address of a PLC and **LSB** is equal to the number of a file. For more information, please check table **File Numbers**
N2	Type of a variable. For more information, please check table **Memory Types**
N3	Number of the first element of a file
N4	Number of the first sub-element of a file. For more information, please check the next **note**

If the default address of a PLC is configured, the *N1* parameter is only the number of a file.

NOTE

To manipulate data with files in **ASCII** format, that is, with the *N2* parameter equal to 31, the *N4* parameter is used to define a data size (**String**), because this type of file does not use addressing with sub-elements. Since internal allocation of data occupies a 16-bit register, only **String** lengths with an even size of characters are allowed.

[B] Parameters for Addressing Block-Type Tags

B1	Address of a PLC (PC source) × 1000 + the number of a file. If a hexadecimal addressing is used, **MSB** is equal to the address of a PLC and **LSB** is equal to the number of a file. For more information, please check table **File Numbers**
B2	Type of a variable. For more information, please check table **Memory Types**
B3	Number of the first element of a file
B4	Number of the first sub-element of a file. For more information, please check the next **note**
Size	Number of Block Elements

If the default address of a PLC is configured, the *B1* parameter is only the number of a file.

NOTE

To manipulate data with files in **ASCII** format, that is, with the *N2* parameter equal to 31, the *N4* parameter is used to define a data size (**String**), because this type of file does not use addressing with sub-elements. Since internal allocation of data occupies a 16-bit register, only **String** lengths with an even size of characters are allowed.

File Numbers

File Numbers

FILE	NUMBER (N1 OR B1)	TYPE	DESCRIPTION
O0	0 (zero)	Output	Status of output terminals
I1	1 (one)	Input	Status of input terminals
S2	2 (two)	Status	Information on the operation of a PLC
B3	3 (three)	Bit	Internal relay logic
T4	4 (four)	Timer	Pre-defined timer, values, and status bits
C5	5 (five)	Counter	Pre-defined counter, values, and status bits
R6	6 (six)	Control	Size, pointer position, and status bits for specific instructions such as register shifting
N7	7 (seven)	Integer	Stores number values and bit information
F8	8 (eight)	Float	Stores a number in a range within 1.1754944E-38 and 3.40282347E+38

FILE	NUMBER (N1 OR B1)	TYPE	DESCRIPTION
F8	8 (eight)	User-defined	User-defined files to store **Bit**, **Timer**, **Counter**, **Control**, or **Integer** values
P9	9 (nine)	PID	PID files that contain several control variables
10-255	10 to 255	User-defined	User-defined files to store **Bit**, **Timer**, **Counter**, **Control**, or **Integer** values
10-255	10 to 255	User-defined	User-defined files to store any **String** (ST) or **ASCII** (A) values

NOTE

Files may vary according to the model of a PLC.

Memory Types

Memory types (variable)

VARIABLE	FILE	TYPE (N2 OR B2)
Word	SLC_OUTPUT	0 (zero)
	SLC_INPUT	1 (one)
	SLC_STATUS	2 (two)
	SLC_BIT	3 (three)
	SLC_TIMER	4 (four)
	SLC_COUNTER	5 (five)
	SLC_CONTROL	6 (six)
	SLC_INTEGER	7 (seven)
	SLC_INTEGER, a signed 16-bit integer	71. For more information, please check the next **note**
BCD	SLC_OUTPUT	8 (eight)
	SLC_INPUT	9 (nine)
	SLC_STATUS	10
	SLC_BIT	11
	SLC_TIMER	12
	SLC_COUNTER	13
	SLC_CONTROL	14
	SLC_INTEGER	15
Bit	SLC_OUTPUT	16, only individual Tags
	SLC_INPUT	17, only individual Tags
	SLC_STATUS	18, only individual Tags

VARIABLE	FILE	TYPE (N2 OR B2)
	SLC_BIT	19, only individual Tags
	SLC_TIMER	20, only individual Tags
	SLC_COUNTER	21, only individual Tags
	SLC_CONTROL	22, only individual Tags
	SLC_INTEGER	23, only individual Tags
Float	SLC_FLOAT	24
Long	SLC_LONG	25
String	SLC_STRING	26, only individual Tags
Bit	SLC_LONG	28
Float	SLC_INTEGER	29
PID	SLC_PID. For more information, please check the next **note**	30
ASCII	SLC_ASCII	31

NOTES

- Memory types 7 (seven) and 71 access the same data in a PLC. The difference between them is how data is interpreted, which in the first case is interpreted as unsigned 16-bit integers (**Word**) and in the second case is interpreted as signed 16-bit integers.
- Blocks for reading PID files must contain a maximum of 23 Elements, addressed from 0 (zero) to 22.

Addressing Examples

Examples of Addressing in the Format N1.N2.N3.N4

- Using a decimal addressing:

```
PLC = 10, F8:40 (10008.24.40.0) (File 8, SLC_FLOAT
PLC = 2, N25:100 (2025.7.100.0) (File 25, SLC_INTEGER)
PLC = 1, C5:42.ACC (1005.5.42.2) (File 5, SLC_COUNTER, Element 3)
PLC = 3, C5:42.PRE (3005.5.42.1) (File 5, SLC_COUNTER, Element 2)
```

- Using a hexadecimal addressing:

```
PLC = 10, F8:40 (1008h.24.40.0) (File 8, SLC_FLOAT)
PLC = 2, N25:100 (0225h.7.100.0) (File 25, SLC_INTEGER)
PLC = 1, C5:42.ACC (0105h.5.42.2) (File 5, SLC_COUNTER, Element 3)
PLC = 3, C5:42.PRE (0305h.5.42.1) (File 5, SLC_COUNTER, Element 2)
```

Examples of Addressing for Files in ASCII format (N2 or B2 equal to 31)

Files in **ASCII** format are addressed by 16-bit elements and their data is manipulated based on an *n* length of data, all in sequence, representing a **String**. For example, to receive 20 characters from address 10 of a file in **ASCII** format, configure a PLC Tag in the way described next.

```
PLC = 1, A9:10 (1009.31.10.20) (File 9, SLC_ASCII, 20 characters from Element 10 of a file)
```

In case of Block Tags, the *n* length of data is processed for each one of its Elements. For example, to receive 20 characters in each of the three Elements of a Block Tag, from the address 10 of a file in **ASCII** format, configure a Block Tag in the way described next.

```
PLC = 1, A9:10 (1009.31.10.20) (File 9, SLC_ASCII, 20 characters from Element 10 of a file, for each
one of the Elements of a Block Tag)
Tag.Element1 = 20 characters from Element 10 of a file
Tag.Element2 = 20 characters from Element 20 of a file
Tag.Element3 = 20 characters from Element 30 of a file
```

NOTES

- Since internal allocation of data occupies a 16-bit register, the configuration in the *N4* or *B4* parameter only allows **String** lengths with an even size of characters.
- **String** readings are conditioned to the length configured in the *N4* or *B4* parameter, but may return a smaller size if a **Null** (0x00) character, or a **String** terminator, is received.
- **String** writings with a length greater than the one configured in the *N4* or *B4* parameter have their characters truncated until the limit of this parameter.
- **String** writings with a length less than the one configured in the *N4* or *B4* parameter are filled with **Null** (0x00) characters to the right, up to the length defined in this parameter.

Tag Configuration via Integers and Strings

This Driver allows configuring Tags using **Integers** and **Strings**. To do so, users must use the **Device** (**ParamDevice**) and **Item** (**ParamItem**) properties, respectively.

The **Device** property of Tags inherits its value from this Driver, can be overwritten in a Tag, and defines the address of a device accessed by a Tag. If this property is left empty, it automatically assumes the value 0 (zero).

The **Item** property, on the other hand, identifies data accessed by a Tag in a device. This property can have one of the following formats, in which values inside brackets are optional.

```
[DATATYPE] AREA FILE  [: ADDRESS] / [BIT]
```

Or for timers and counters:

```
[DATATYPE] AREA FILE  : ADDRESS . SUBELEMENT
```

Or for files in **ASCII** format:

```
[DATATYPE] AREA FILE  : ADDRESS - DATA SIZE (STRING)
```

The next tables define all mnemonics that can be used in the previously described values.

Types of variables

TYPE OF VARIABLE	STRING	N2 OR B2 PARAMETER
Word	W	Between 0 (zero) and 7 (seven)
Long	DW, used only with the **L** area	25
Signed Word	SW, used only with **I** integer types	71
BCD	BCD	Between 8 (eight) and 15
Bit	Whenever the optional **Bit** value is defined	Between 16 and 23
Float	Not defined. To use this type of variable, define an area as **F**	24
String	Not defined. To use this type of variable, define an area as **ST**	26

TYPE OF VARIABLE	STRING	N2 OR B2 PARAMETER
ASCII	Not defined. To use this type of variable, define an area as **A**	31

Areas

AREA	STRING
SLC_OUTPUT	O
SLC_INPUT	I
SLC_STATUS	S
SLC_BIT	B
SLC_TIMER	T
SLC_COUNTER	C
SLC_CONTROL	R
SLC_INTEGER	N
SLC_FLOAT	F
SLC_LONG	L, always with a **DW**-type of variable
SLC_STRING	ST
SLC_ASCII	A
SLC_PID	PD, always with a **W**-type of variable

The next table shows examples of Tags configured by **Strings**, linking them with their equivalent *N* or *B* parameters.

Examples of Tags defined by Strings

DEVICE	ITEM	N1 OR B1	N1 OR B1 (HEX)	N2 OR B2	N3 OR B3	N4 OR B4
2 (two)	WO0:0/4	2000	0200h	16	0 (zero)	4 (four)
8 (eight)	SWI1:86	8001	0801h	71	86	0 (zero)
8 (eight)	WS2:32	8002	0802h	2 (two)	32	0 (zero)
0 (zero)	WI1:104	1 (one)	1 (one)	1 (one)	104	0 (zero)
11	BCDT4:38.9	11004	0B04h	12	38	9 (nine)
10	F8:3	10008	0A08h	24	3 (three)	0 (zero)
6 (six)	ST4:30	6004	0604h	26	30	0 (zero)
13	L7:90	13007	0D07h	25	90	0 (zero)
13	DWL7:100	13007	0D07h	25	100	0 (zero)
11	PD4:5.15	11004	0B04h	30	5 (five)	15
3 (three)	A9:10-20	3009	0309h	31	10	20

Troubleshooting

1. When trying to communicate with a device, users must be sure that it is not communicating with another application. It is common that users configure a device with RS Linx and then do not deactivate its communication with that PLC when finished communicating.

2. In case of trying to read or write a non-existent element, the system indicates a communication error. This also happens in case of trying to read a Block with more **Words** than the specified file contains.

3. Users can only configure a Tag to communicate with a bit if the sub-element of a file is equal to 0 (zero), because when configuring the *N2* or *B2* parameter to access that bit, this Driver considers that this sub-element is equal to 0 (zero). If users want to retrieve these bits and the sub-element is different from 0 (zero), it is advisable to configure a Tag to retrieve the entire data and then access these bits with the existing **Elipse E3**, **Elipse Power**, **Elipse Water**, or **Elipse SCADA** functions.

4. When trying to read or write directly to inputs and outputs, that is, to **O0** and **I1** files, depending on the model and version of a device, there may have some problems in the number format for addressing, generating communication errors that are registered in the log with a message "AB_command returned error in status = 10!". It is advisable to map inputs and outputs to integers, as described in the article *Address issues when accessing input and output files (O0 and I1) with Allen-Bradley ABDF1 Driver* on **Elipse Knowledgebase**.

5. When trying to communicate with a MicroLogix 1100 device using a local Ethernet port and communication fails, change the **PLC Type** configuration to **MicroLogix 1500**. Some firmware versions of that device require this configuration on this Driver.

Superblock Reading

This Driver supports Superblocks since version **1.7**. This feature allows **Elipse E3**, **Elipse Power**, or **Elipse Water** to group Tags configured in an application into blocks with the largest possible size, to optimize communication. To enable this feature in **Elipse E3**, in **Elipse Power**, or in **Elipse Water**, configure the **EnableReadGrouping** property of a Driver object to True.

Not all data types support this feature. **BCD** data types and **Counter** and **Timer** areas cannot be grouped.

Support for grouping blocks of bits is not yet implemented in the current version of this Driver, but it may be implemented in future versions.

Data type **29**, which reads 16-bit integers and formats them as 32-bit **Floats**, also cannot be grouped in the current version of this Driver.

For Tags with outputs in **String** format, it is recommended to use **A**, or **ASCII**, tables for grouping **String** readings via Superblocks. If a project opts by using **ST**, or **String**, tables, group readings are not allowed for this type of table.

Legacy Mode

To keep compatibility with older applications, a new operation mode called **Legacy Mode** was created in version **1.15** of this Driver. In this mode, this Driver behaves as in version **1.14**. This mode can be enabled or disabled as follows:

- When including a new Driver in an application and loading the file ABDF1.dll, this mode is disabled
- If there is already an earlier version of this Driver in an application and the file ABDF1.dll is replaced by a later version, this mode is enabled

In addition, there is an option to enable or disable it at run time using the **Set Configuration Parameters** Tag of **IOKit** library. The parameter to configure is *ABDF1.LegacyMode*, and its value must be 0 (zero) to disable it and 1 (one) to enable it.

For more information about the usage of the **Set Configuration Parameters** Tag of **IOKit** library, please check topic **Documentation of I/O Interfaces - General Configurations - I/O Tags**.

For more information about differences among versions of this Driver, please check topic **Driver Revision History**.

Documentation of I/O Interfaces

This section contains the documentation of I/O Interfaces referring to the **ABDF1** 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.

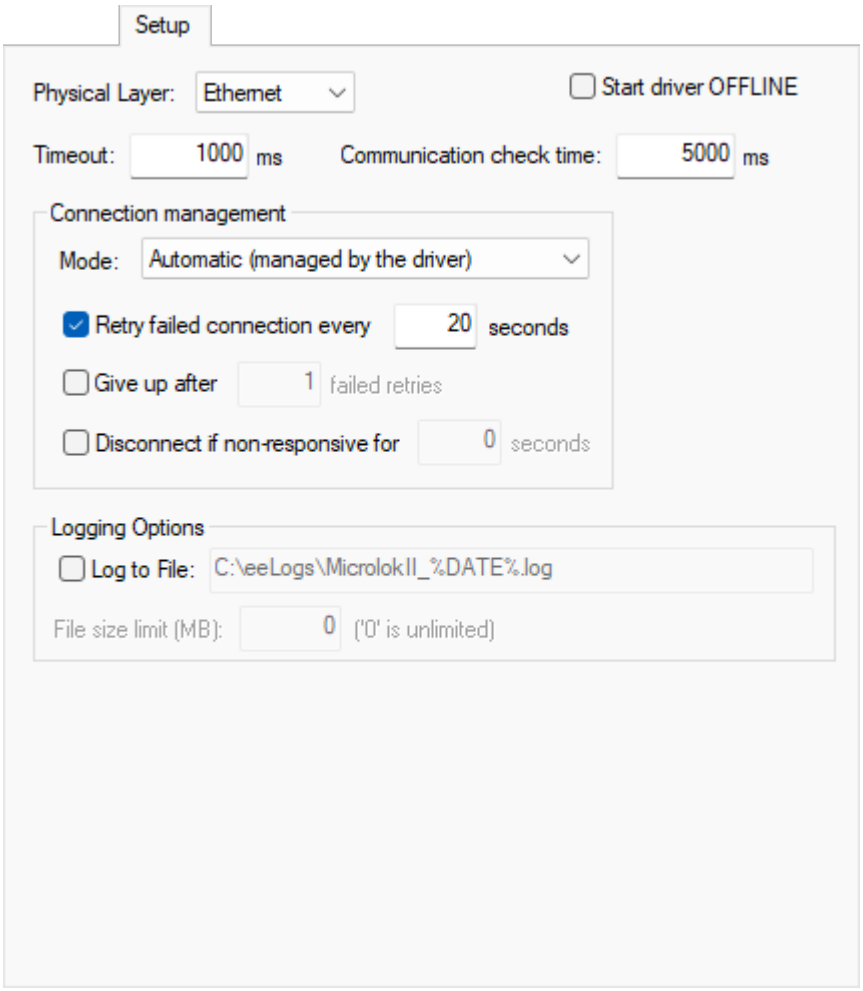
Configuration Dialog Box

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

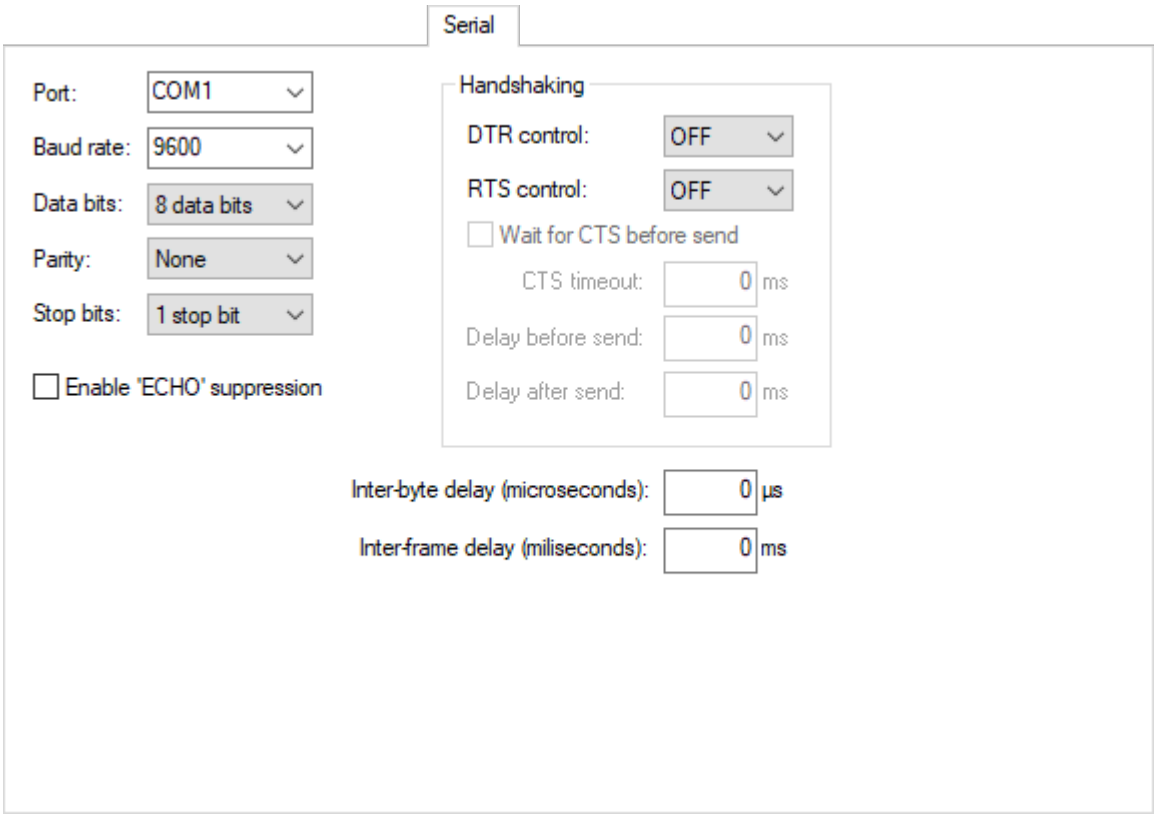
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection
Retry failed connection every ... 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

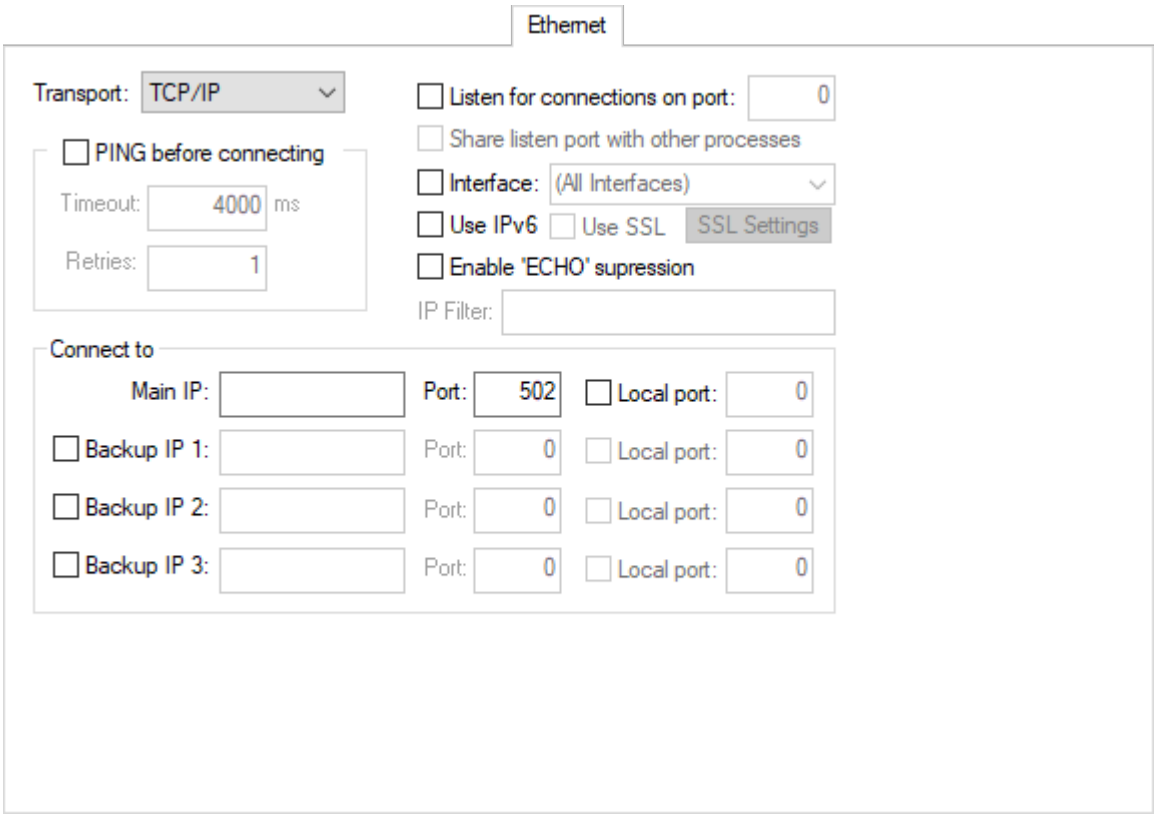
The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process, that is, it controls when data can be sent or received via serial line. Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works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

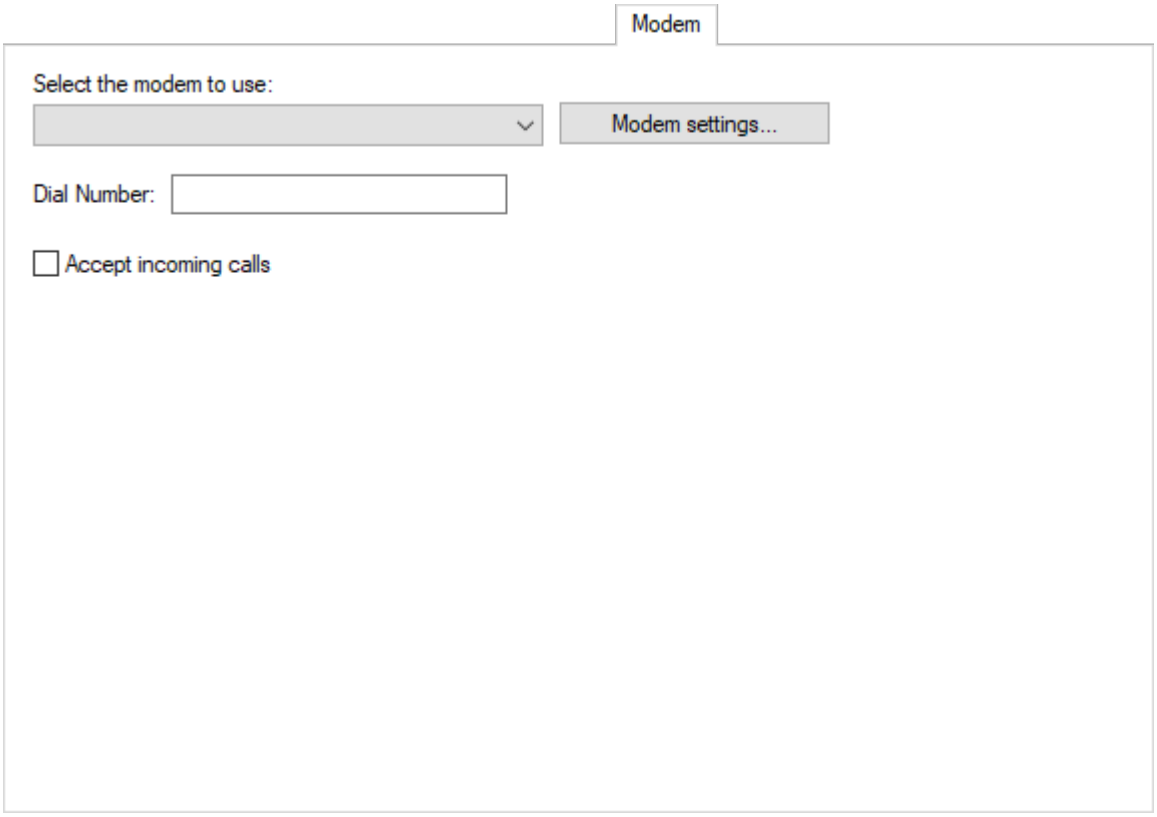
OPTION	DESCRIPTION
	of determining a successful connection before trying to open a socket with a device. The time-out of a connection with a socket can be very high. The available options are: • **Timeout**: Specify the number of milliseconds to wait for a reply from a **ping** command. Users must use 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.

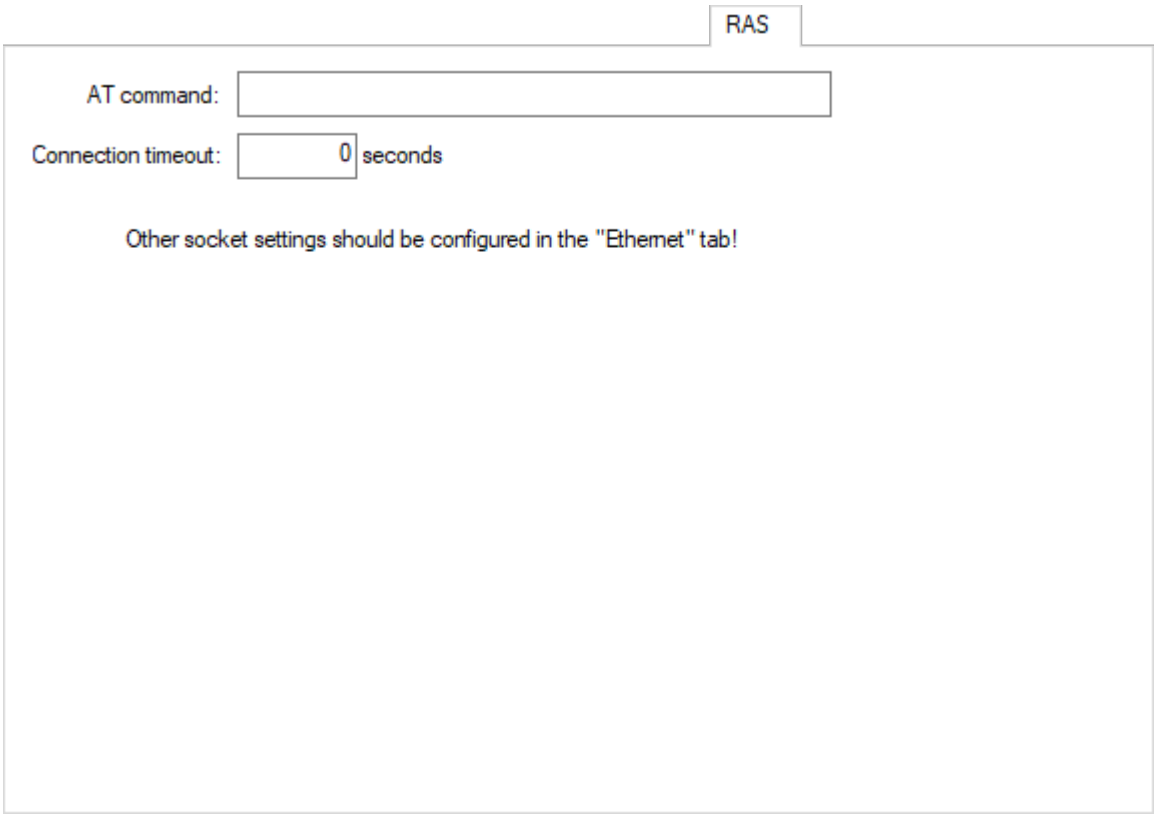
A **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

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

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

IO.TAPI.ModemID

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

NOTE

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

IO.TAPI.PhoneNumber

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

RAS Interface Configuration

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

I/O Tags

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

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

Properties

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

NOTE

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

IO.RAS.ATCommand

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

IO.RAS.CommandTimeoutSec

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

Serial Interface Configuration

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

I/O Tags

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

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

Properties

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

IO.Serial.Baudrate

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

IO.Serial.CTSTimeoutMs

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

IO.Serial.DataBits

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

IO.Serial.DelayAfterMs

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

IO.Serial.DelayBeforeMs

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

IO.Serial.DTR

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

IO.Serial.InterbyteDelayUs

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

IO.Serial.InterframeDelayMs

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

IO.Serial.Parity

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

IO.Serial.Port

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

IO.Serial.RTS

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

IO.Serial.StopBits

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

IO.Serial.SupressEcho

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

IO.Serial.WaitCTS

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

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
2.0.18	08/22/2025	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 37930*).
2.0.17	12/08/2023	P. H. Santos	• Implemented a validation of Tags and assumed the value 0 (zero) for the **Device** property if that property is empty (*Case 35216*).
2.0.16	19/08/2019	C. Mello	• Platform update on this Driver's source code (*Case 27354*).
2.0.15	03/07/2019	C. Mello	• Added support for Superblock readings in files in **ASCII** format (*Case 24440*).

VERSION	DATE	AUTHOR	COMMENTS
2.0.14	01/28/2019	C. Mello	• Added support for using Block Tags in files in **ASCII** format (*Case 26033*).
2.0.13	09/17/2018	C. Mello	• Added support for manipulating data in files in **ASCII** format (*Case 25046*).
2.0.10	02/13/2017	A. Hertzog	• Added support for **PID** files (*Case 21560*).
2.0.6	04/15/2016	M. Salvador	• Implemented support for direct writings and readings of bits in **SLC_BIT** areas (*Case 20093*).
2.0.5	09/10/2015	A. Quites	• Added an advanced option to optimize **String** readings (*Case 18436*).
2.0.1	09/26/2013	G. Taschetto	• Driver ported to **IOKit** library version **2.0** (*Case 14126*).
1.15.1	11/21/2012	A. Quites G. Taschetto	• Implemented a transaction control for DF1 protocol encapsulated in ENIP, allowing to reject any delayed response frames (*Case 13090*). • Grouping by Superblocks was disabled for Tags accessing input and output files (the *N1* or *B1* parameter equal to zero or equal to one), to avoid problems with an eventual grouping of non-adjacent cards (*Case 13084*). • Added a suggestion of mapping of digital inputs and outputs for integer variables on topic **Troubleshooting** (*Case 12243*). • Fixed a problem in which users could not write to bits above 15 in I/O cards with 32 or more bits (*Case 13274*). • Added a protection on the configuration of Tags by **Strings** against the use of bits in **Float** data types and sub-elements in **Integer** and **Float** data types (*Case 13339*).

VERSION	DATE	AUTHOR	COMMENTS
			• Added support for I/O cards up to 64 bits (*Case 13330*).
1.14.1	12/01/2010	A. Quites	• Fixed an error when writing bits of **Long** data type variables (*Case 11923*).
1.13.1	10/13/2010	A. Quites	• Fixed an error when writing bits of a most significant **Word** data type of 32-bit data types (*Case 10768*). • Added a transaction check (TNS) of DF1 layer when encapsulated in CSPv4 (*Case 11023*). • Fixed an error when writing Block Elements with a **Float** data type (*Case 11024*). • Driver ported to **Windows CE** (*Case 10913*). • Fixed an error when reading a **71** (**I16**) data type in **Elipse E3**, with Superblocks (the **EnableReadGrouping** property) disabled (*Case 11653*). • Fixed all addressing examples of sub-element **ACC** (*Case 10979*).
1.12.1	04/22/2009	A. Quites	• Fixed an error when reading internal Tags of **IOKit** library (*Case 10242*). • Fixed an offset error when writing to Block Elements with 32-bit data types (*Case 10253*). • Fixed an error when writing to bits from 16 to 31 in 32-bit data types (*Case 10345*). • Implemented a new **29** data type, which reads two **Words** and converts them to a **Float** (*Case 10016*). • Fixed an offset error when reading Blocks with 32-bit data types with a total size greater than the maximum configured limit (*Case 10296*). • Support for Superblocks with 32-bit data types disabled, due to an error

VERSION	DATE	AUTHOR	COMMENTS
			with 16-bit addressing (*Case 10297*).
1.11.1	10/07/2008	M. Salvador A. Quites	• Added support for Micrologix 1500 B series devices (*Case 9132*). • Added support for writing bits in **Long** data types (*Case 9743*). • Added a signed integer data type, corresponding to the *N2* parameter equal to 71 (*Case 9800*).
1.10.1	02/28/2007	A. Quites	• Fixed an error with synchronous TNS (*Case 7833*).
1.9.1	10/10/2006	M. Salvador	• Added support for Micrologix 1100 devices (*Case 7454*).
1.8.1	08/28/2006	A. Quites	• Fixed an error when receiving packets in **BCC** (*Case 7289*).
1.7.1	07/26/2006	A. Quites M. Salvador	• Implemented features of Superblocks and configuration of Tags by **Strings** (*Case 6742*). • Fixed an error when reading and writing input and output bits (*Case 6415*). • Fixed a problem with the reception in **Half-Duplex** and **Full-Duplex** modes.
1.6.1	04/18/2004	M. Salvador	• Implemented a control for the maximum number of bytes in reception. • Implemented the reading of blocks greater than the maximum number of bytes. • Implemented bit writings.
1.5.1 Beta	04/15/2004	A. Quites M. Salvador	• Implemented a **Half-Duplex** mode on the **DF1** layer (*Case 5160*). • Created a **Use hexadecimal addressing on N1/B1** option.
1.2.1	06/24/2004	M. Salvador	• Version previous to version control.
1.0.1	05/31/2004	M. Salvador	• Original version of this Driver (*Case 1020*).

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)