Allen Bradley DF1 Driver

Filename	ABDF1.dll
Manufacturer	Rockwell
Devices	PLC5, SLC500, MicroLogix, and ControlLogix
Protocol	DF1, DF1 over CSPv4, PCCC, and Ethernet/IP
Version	2.0.16
Last Update	08/19/2019
Platform	Win32
Dependencies	IOKit v2.00 or later
Superblock readings	Yes
Level	0

Introduction

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

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

Preparing the Equipment

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

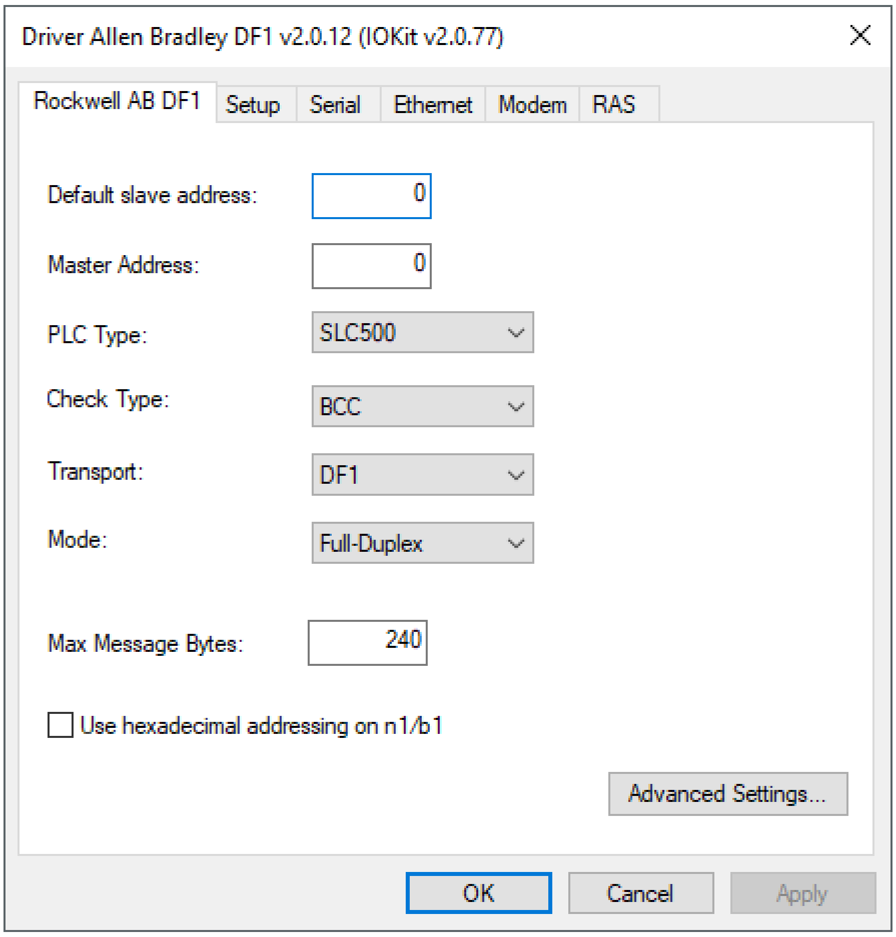
[P] Parameters for Driver Configuration

PARAMETER	VALUE
P1	Not used
P2	Not used
P3	Not used
P4	Not used

The Allen Bradley DF1 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.


Driver Allen Bradley DF1 v2.0.12 (IOKit v2.0.77)

Default slave address: 0

Master Address: 0

PLC Type: SLC500

Check Type: BCC

Transport: DF1

Mode: Full-Duplex

Max Message Bytes: 240

☐ Use hexadecimal addressing on n1/b1

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

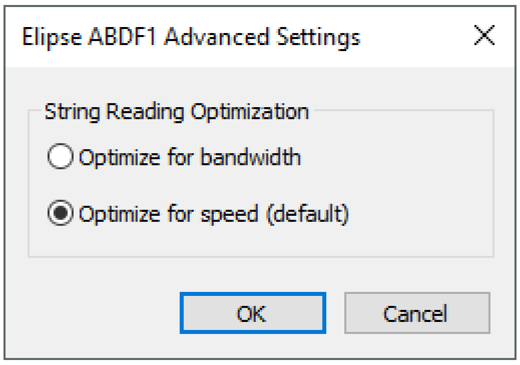
OPTION	DESCRIPTION
Default slave address	Default PLC address. Used whenever Tag addressing does not inform an address. For **Ethernet CSP** and **Ethernet/IP (ENIP)** communication, address is always 0 (zero)
Master Address	Default Master address. For **Ethernet** communication, address is 0 (zero)
PLC Type	Type of 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

OPTION	DESCRIPTION
Mode	Inform whether **Half-duplex** or **Full-duplex** mode is used. This option is only relevant if the **DF1** format is used as a transport layer
Max Message Bytes	This parameter defines the maximum number of bytes of 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 by this option. Default value is 240. Check device's documentation if this value can be increased. If still in doubt, use the default value
Use hexadecimal addressing on n1/b1	If this option is enabled, Tag's *N1* and *B1* parameters are changed using an addressing in hexadecimal format. The most significant byte (MSB) indicates the 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/B1* parameters always in hexadecimal format, adding an "h" character in **Elipse SCADA** or the prefix "&H" in **E3** (for more information, please check *VBScript Language Reference*). Leave this option disabled 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** tabs. For serial communication, 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 **Driver's Configuration Window**.



String Reading Optimization

Optimize for bandwidth [✓ Optimize for speed (default)]

Advanced settings window

This window gathers rarely used settings, which usually do not require user 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 two requests to a device. The first request only reads a numerical value that indicates the real size of the **String** and the second request reads that **String** with the 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 DLLs 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 character) and later discarding unused bytes. This method tends to double the speed of the previous option, but it may require reading more characters in this single reading (always 82 characters) than needed

NOTE

When updating a Driver's DLL in instances initially created with versions earlier than 2.0.3 in old 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

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

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

NOTE

To manipulate data with **ASCII** files, that is, with the *N2* parameter equal to 31, the *N4* parameter is used to define data size (**String**), because this type of file does not use addressing with subelements. 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

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

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

NOTE

To manipulate data with **ASCII** files, that is, with the *N2* parameter equal to 31, the *N4* parameter is used to define data size (**String**), because this type of file does not use addressing with subelements. 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/B1)	TYPE	DESCRIPTION
O0	0 (zero)	Output	Status of output terminals
I1	1 (one)	Input	Status of input terminals
S2	2 (two)	Status	PLC's information on operation
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 shifts
N7	7 (seven)	Integer	Stores numerical values and bit information
F8	8 (eight)	Float	Stores a number in a range within 1.1754944E-38 and 3.40282347E+38
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 many 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 a PLC's model.

Memory Types

Memory types (variable)

VARIABLE	FILE	TYPE (N2/B2)
Word	SLC_OUTPUT	0 (zero)
	SLC_INPUT	1 (one)
	SLC_STATUS	2 (two)

VARIABLE	FILE	TYPE (N2/B2)
	SLC_BIT	3 (three)
	SLC_TIMER	4 (four)
	SLC_COUNTER	5 (five)
	SLC_CONTROL	6 (six)
	SLC_INTEGER	7 (seven)
	SLC_INTEGER (signed 16-bit integer)	71 (please check the **Note** later)
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 PLC Tags)
	SLC_INPUT	17 (only individual PLC Tags)
	SLC_STATUS	18 (only individual PLC Tags)
	SLC_BIT	19 (only individual PLC Tags)
	SLC_TIMER	20 (only individual PLC Tags)
	SLC_COUNTER	21 (only individual PLC Tags)
	SLC_CONTROL	22 (only individual PLC Tags)
	SLC_INTEGER	23 (only individual PLC Tags)
Float	SLC_FLOAT	24
Long	SLC_LONG	25
String	SLC_STRING	26 (only individual PLC Tags)
Bit	SLC_LONG	28
Float	SLC_INTEGER	29
PID	SLC_PID (please check the **Note** later)	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.
- PID file reading blocks must contem 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 ASCII Files (N2/B2 = 31)

ASCII files 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 an ASCII file, configure a PLC Tag in the following way:

```
CLP = 1, A9:10 (1009.31.10.20) (File 9, SLC_ASCII, 20 characters from file's Element 10)
```

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 an ASCII file, configure a Block Tag in the following way:

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

NOTES

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

Tag Configuration via Strings

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

Tag's **Device** property inherits its value from the Driver, and it can be overwritten in the Tag, which defines a device's address accessed by a Tag.

The **Item** property identifies data accessed by a Tag in a device. The **Item** property can have one of the following formats (fields inside brackets are optional).

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

Or for timers and counters:

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

Or for **ASCII** files:

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

The next tables show all mnemonics that can be used in the previously described fields.

Types of variables

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

AREA	STRING
SLC_INTEGER	N
SLC_FLOAT	F
SLC_LONG	L (always a **DW** type)
SLC_STRING	ST
SLC_ASCII	A
SLC_PID	PD (always a **W** type)

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

Examples of Tags defined by Strings

DEVICE	ITEM	N1/B1	N1/B1 (HEX)	N2/B2	N3/B3	N4/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	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 device when finishing their work.

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 a specified file contains.

3. Users can 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/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 this Tag to retrieve the entire data and then access these bits with the existing **E3** or **Elipse SCADA** functions.

4. When trying to read or write directly to inputs and outputs (**O0** and **I1** files), depending on device's model and version, there may have some problems in the numerical addressing format, 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 using a local Ethernet port and communication fails, try to 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 1.07. This feature allows **E3** to group Tags configured in an application into blocks with the largest possible size, to optimize communication. To enable this feature in **E3**, users must configure I/O Driver's **EnableReadGrouping** property 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 this Driver's current version, 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 this Driver's current version.

For Tags with outputs in **String** format, it is recommended to use **A**, or **ASCII**, tables for grouping **String** readings via Superblocks. If the project uses **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. 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 via **IOKit**'s **Set Configuration Parameters** Tag. 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 **IOKit**'s **Set Configuration Parameters** Tag, please check topic **Documentation of I/O Interfaces - General Configurations - I/O Tags**.

For more information about differences among version 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 **Allen Bradley DF1** 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.

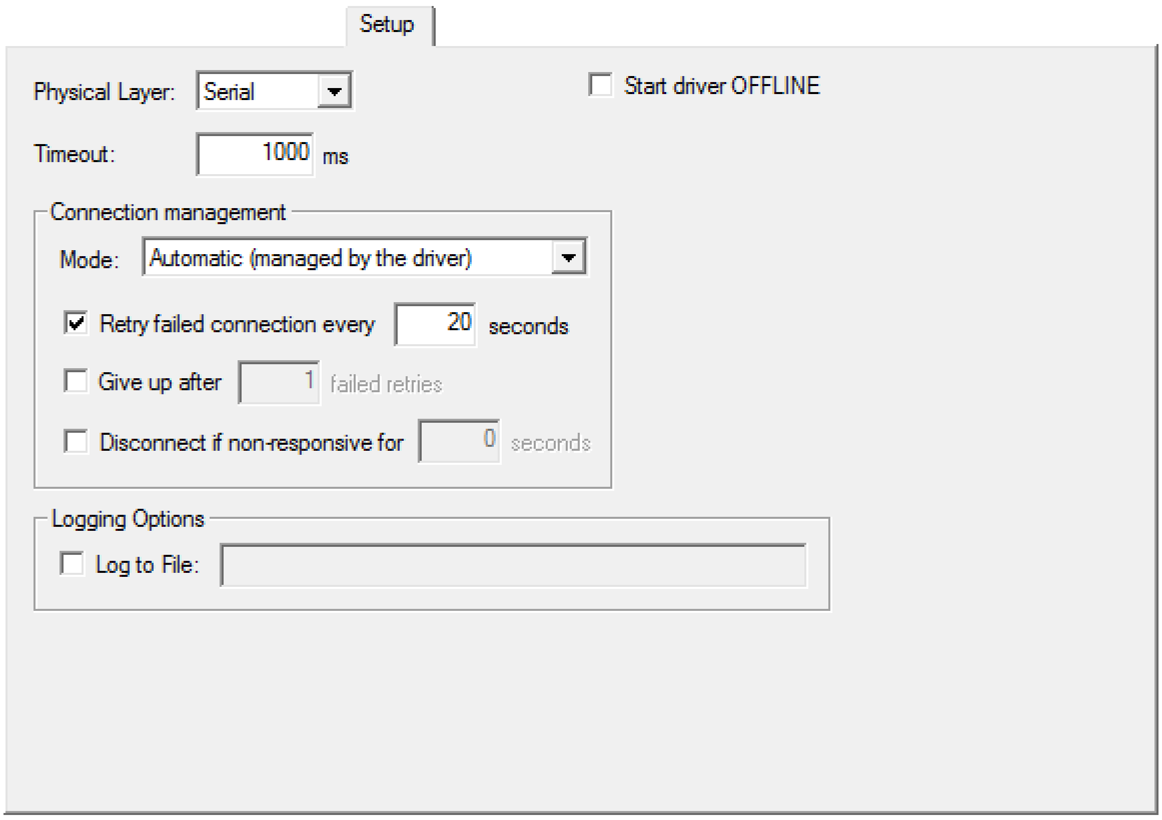
Configuration Dialog Box

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

Physical Layer: Serial

Start driver OFFLINE ☐

Timeout: 1000 ms

Connection management

Mode: Automatic (managed by the driver)

☑ Retry failed connection every 20 seconds

☐ Give up after 1 failed retries

☐ Disconnect if non-responsive for 0 seconds

Logging Options

☐ Log to File:

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

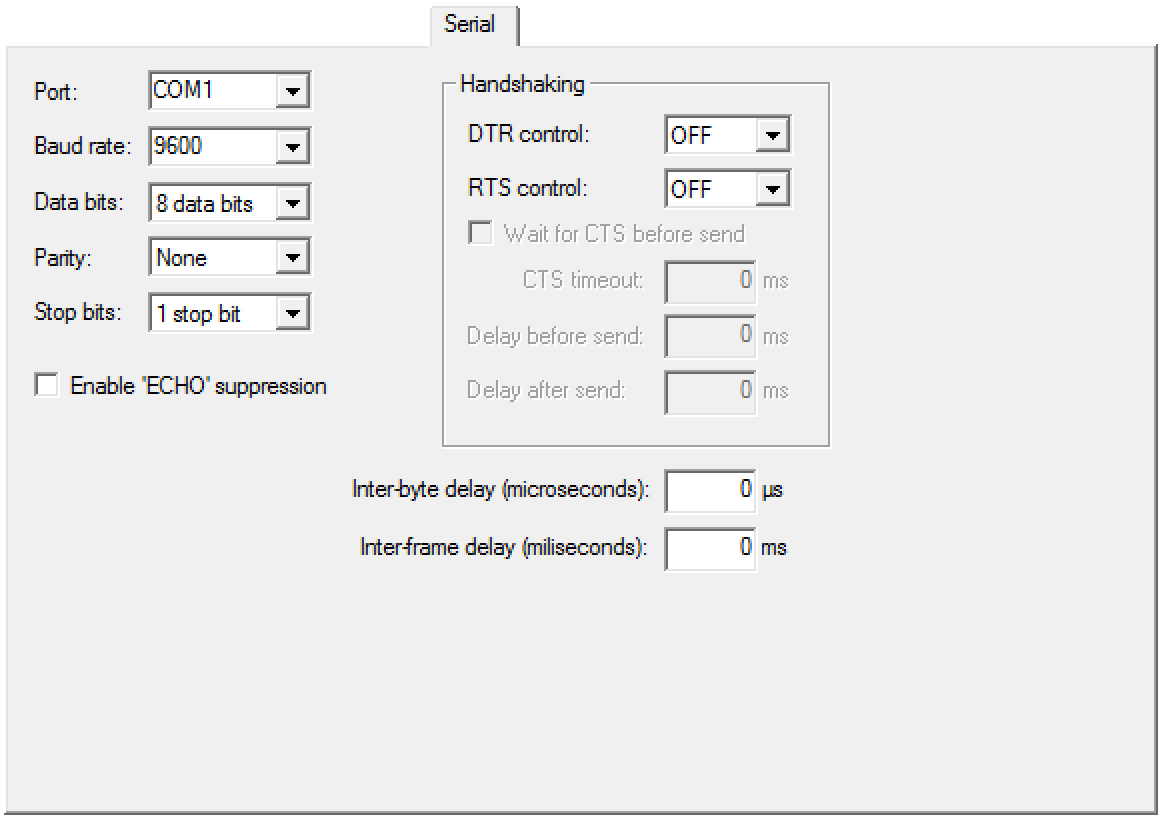
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage the connection. Please check topic **Driver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

Port: COM1
Baud rate: 9600
Data bits: 8 data bits
Parity: None
Stop bits: 1 stop bit
Enable 'ECHO' suppression

Handshaking
DTR control: OFF
RTS control: OFF
Wait for CTS before send
CTS timeout: 0 ms
Delay before send: 0 ms
Delay after send: 0 ms

Inter-byte delay (microseconds): 0 µs
Inter-frame delay (miliseconds): 0 ms

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.

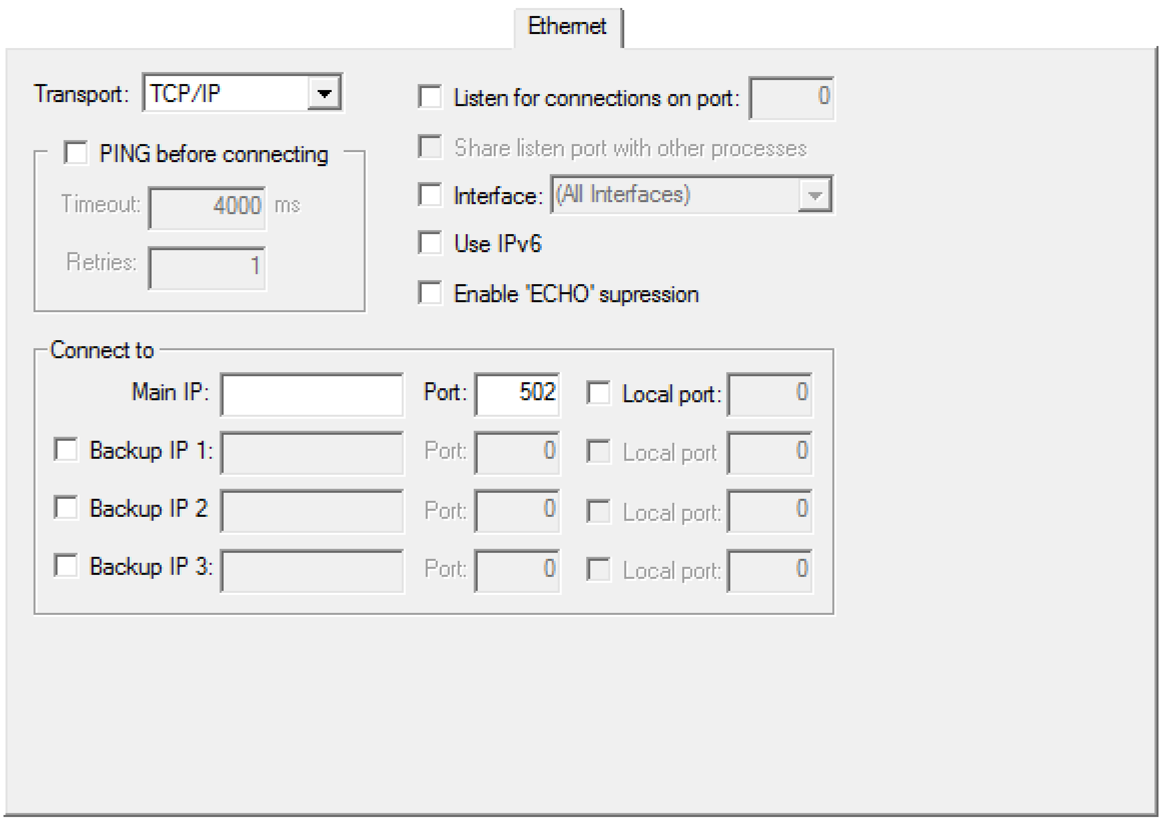
The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process (controlling when data can be sent or received via serial line). Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works with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



Transport: TCP/IP
PING before connecting
Timeout: 4000 ms
Retries: 1
Listen for connections on port: 0
Share listen port with other processes
Interface: (All Interfaces)
Use IPv6
Enable 'ECHO' supression
Connect to
Main IP: Port: 502 Local port: 0
Backup IP 1: Port: 0 Local port 0
Backup IP 2 Port: 0 Local port: 0
Backup IP 3: Port: 0 Local port: 0

Ethernet tab

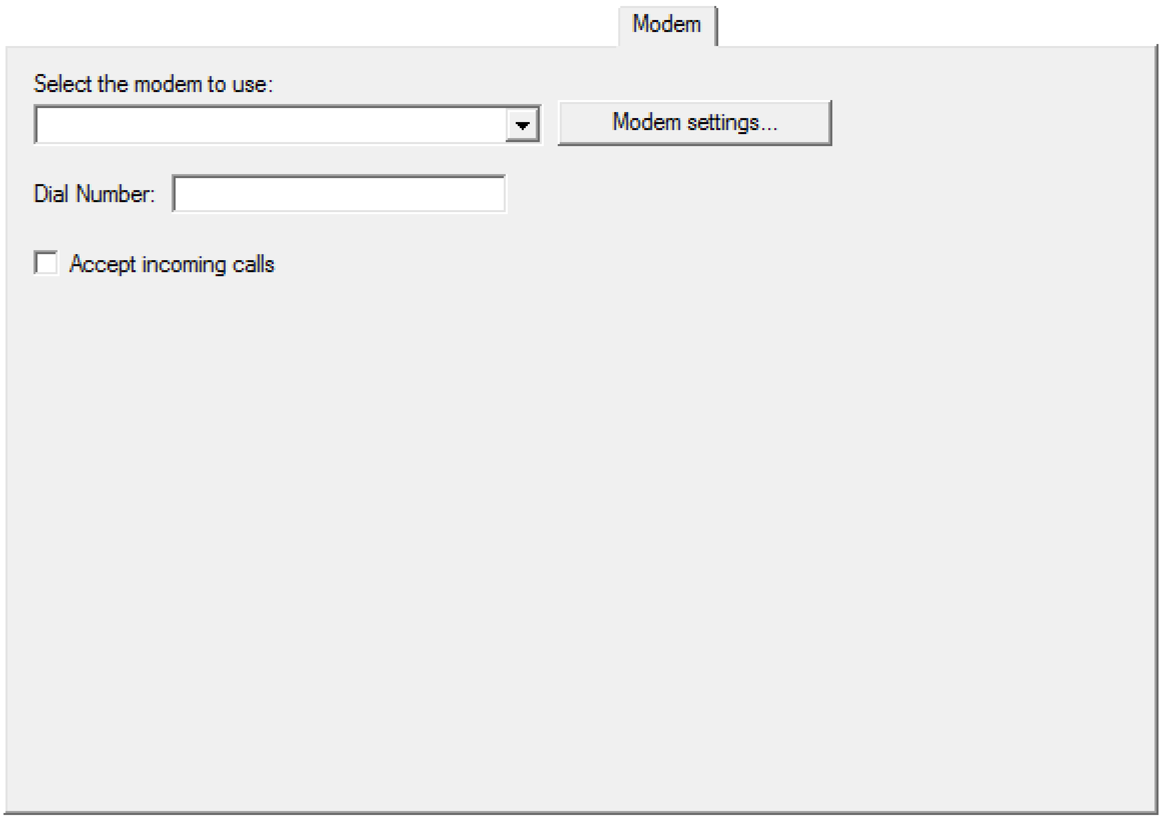
Available options on Ethernet tab

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.

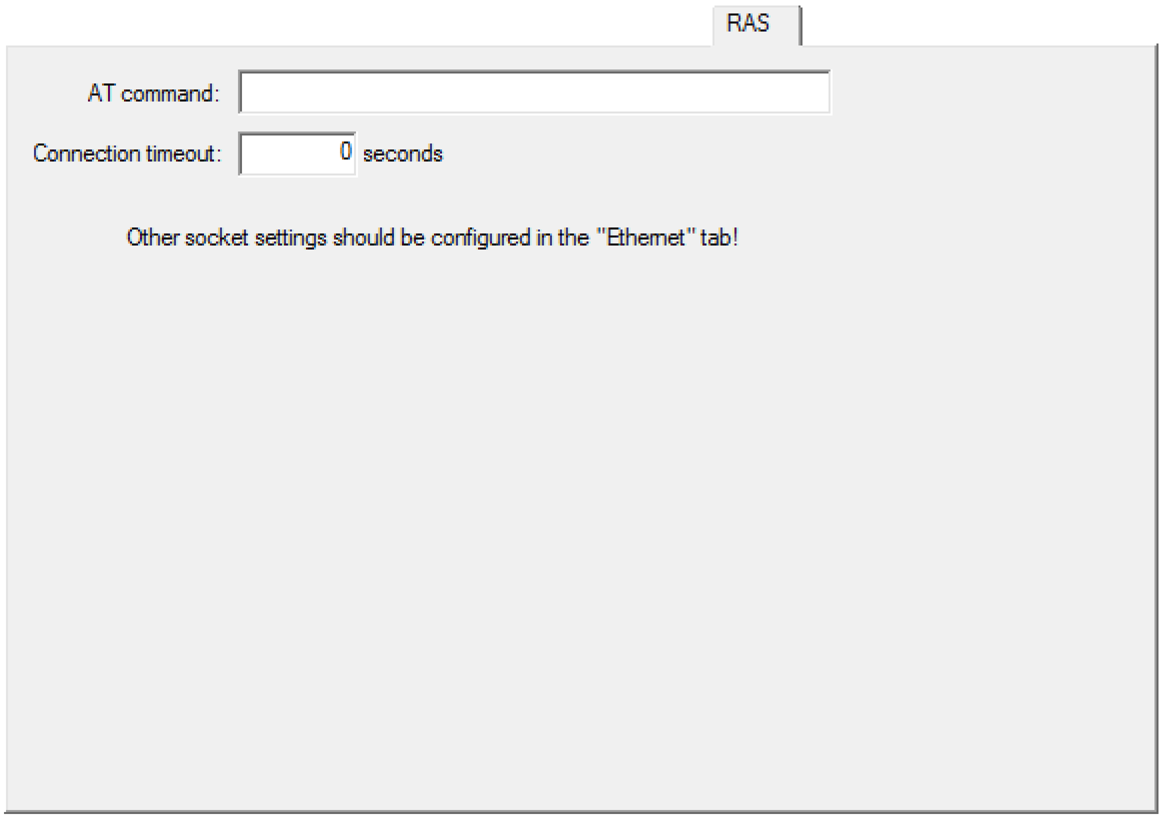
The **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

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.



AT command:

Connection timeout: 0 seconds

Other socket settings should be configured in the "Ethernet" 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

A 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

A 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

A 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

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

IO.Ethernet.MainIP

A 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

A **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

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 **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

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

IO.Serial.CTSTimeoutMs

9 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

9 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

A 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

9 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

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 to **Toggle**.

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
2.0.16	08/19/2019	C. Mello	• Driver source code platform update (*Case 27354*).
2.0.15	03/07/2019	C. Mello	• Added support for Superblock readings for files in **ASCII** format (*Case 24440*).
2.0.14	01/28/2019	C. Mello	• Added support for using Block Tags in files in **ASCII** format (*Case 26033*).

VERSION	DATE	AUTHOR	COMMENTS
2.0.13	09/17/2018	C. Mello	• Added support for data manipulation 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** v2.0 (*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 (*N1/B1* equal to zero or *N1/B1* 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*). • Added support for I/O cards up to 64 bits (*Case 13330*).

VERSION	DATE	AUTHOR	COMMENTS
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 of **Float** data type (*Case 11024*). • Driver ported to **Windows CE** (*Case 10913*). • Fixed an error when reading a **71** (**I16**) data type in **E3**, with Superblocks (**EnableReadGrouping**) 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 **IOKit**'s internal Tags (*Case 10242*). • Fixed a shifting error (offset) when writing to Block Elements of 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 a shifting error (offset) when reading Blocks of 32-bit data types with a total size greater than the maximum configured limit (*Case 10296*). • Support for Superblocks of 32-bit data types disabled, due to an error with 16-bit addressing (*Case 10297*).

VERSION	DATE	AUTHOR	COMMENTS
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, *N2* equal to 71 (*Case 9800*).
1.10.1	02/28/2007	A. Quites	• Fixed a problem with a synchronous TNS error (*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 reception in **Half-Duplex** and **Full-Duplex** modes.
1.06	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.
1.05B	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.
1.02	06/24/2004	M. Salvador	• Version previous to version control.
1.00	05/31/2004	M. Salvador	• Driver's original version (*Case 1020*).

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 CoolVendor

OPC FOUNDATION MEMBER

Microsoft Partner
Gold Independent Software Vendor (ISV)