

ASCII Generic Driver

Filename	ASCII.dll
Manufacturer	Eclipse Software
Devices	Any device with an ASCII I/O interface
Protocol	Any ASCII protocol
Version	2.0.5
Last Update	06/13/2025
Platform	Win32
Dependencies	IOKit version 2.0 or later
Superblock Readings	No
Level	0

Introduction

The ASCII Generic Driver communicates with any device using the **ASCII** format.

Driver Configurations

This Driver's **[P]** parameters are not used. All configurations are performed on this Driver's configuration dialog box. For more information about **Serial**, **Ethernet**, **Modem**, and **RAS** tabs, please check topic **Documentation of I/O Interfaces**.

Configuring Properties

The **ASCII** tab contains specific configurations for this Driver. The available options on this tab are described on the next table.

Available options on the ASCII tab

OPTION	DESCRIPTION
Max Answer Bytes	Maximum number of bytes expected in the response
Max Answer Elements	Maximum number of variables, or elements, in a command or response
ID	Index of a command linked to the <i>N2/B2</i> parameter of a PLC or Block Tag
TX	Format of a send command. For more information, please check topic Configuring ASCII Commands
RX	Format of a receive command. For more information, please check topic Configuring ASCII Commands

The **ID**, **TX**, and **RX** options are items that compose a list of user-defined commands. Fill in these options of a command and add it to the list of commands by clicking **Add**. To change them, select a command on the list and use the **Update** option to change it or **Del** to delete it.

Configuring ASCII Commands

ASCII commands are assembled to send or extract data from I/O packets according to a device's protocol. For each format for sending and receiving, an index or ID is associated. A command can have any numbers or letters in **ASCII** format, fixed or with variable fields. To create variable fields, the general rule is to use the **%Nt** format, as described on the next table.

Available formats for ASCII commands

FORMAT	DESCRIPTION
%	Symbol identifying a variable field. If a message must send a % (percentage) character, use the value "%%"
N (optional)	Number of characters or formatter. If a field is a real number, N can be a real number, such as 5.2, which indicates five or more numbers on the integer part plus two numbers on the fractional part
t	Data type. Possible values are d : Signed decimal, u : Unsigned decimal, x : Hexadecimal, f : Real number, b : Binary (character's binary value), s : String (characters of any type), t : Text (only characters that can be printed, that is, whose ASCII code is greater or equal to 32 or 20h), or n : Interprets a decimal value in BCD format

NOTE

For **s (String)** and **t** (text) data types, there are subtypes **@** and **#**, which can be used when users want to extract only a portion of a **String** or a text:

- Use the **@** symbol to identify a special character as a **String**'s initial delimiter
- Use the **#** symbol to identify a special character as a **String**'s final delimiter. Users can also define this special character in hexadecimal notation, by using the **h** symbol followed by two more characters representing its ASCII character in hexadecimal
- Use the **%s** format to extract a full **String**
- Use the **%@g#;s** format to extract a substring from the **g** character to the **;** (semicolon) character
- Use the **%@h02#h03s** format to extract a substring from the **0x02** character to the **0x03** character
- Delimiter characters of symbols **@** and **#** are not part of data extraction representing a substring of **%s** format

Users can also insert or check special characters **/r** (*carriage return*), **/n** (*line feed*), and any other special character in the format **/hXX**, by inserting characters **/** (slash mark) or **** (backslash) followed by **r**, **n**, or **h** directly on the **TX** or **RX** message. Insertion or comparison is performed directly with special characters **/r** (13), **/n** (10), or **/hXX**, in which **XX** represents a special character in hexadecimal format with two digits.

Users can also define which byte is the initial byte of data reception, which byte is the final byte of data reception, and whether these bytes must be part of data extraction, according to the next options:

- **[xx]**: Defines which byte starts data reception, in which **xx** is in hexadecimal notation and included in data extraction
- **]xx**: Defines which byte ends data reception, in which **xx** is in hexadecimal notation and included in data extraction
- **(xx)**: Defines which byte starts data reception, in which **xx** is in hexadecimal notation and not included in data extraction

- **)xx**: Defines which byte ends data reception, in which **xx** is in hexadecimal notation and not included in data extraction

In addition, for **RX** responses that use the **]xx** or **)xx** terminator bytes and if there is a need to discard a few bytes that follow immediately, users can add a parameter to indicate the number of bytes that must be discarded by this Driver after receiving this byte terminator, aiming to clear data's reception buffer.

To use this parameter to discard the last *n* bytes, use the expression `"*nn"` right after the parameter that defines data reception's terminator byte, that is, **]xx*nn** or **)xx*nn**, in which **nn** is in decimal notation and represents the number of bytes received and discarded by this Driver.

The next tables contain examples of usage for ASCII commands.

Usage examples

PROTOCOL	GOAL	CONFIGURATION
TX: <STX>query<ETX> RX: <STX>1200<ETX>	Send a query command to a device and extract a response's numerical value	<ul style="list-style-type: none"> • ID: 1 (one) • TX: /h02query/h03 • RX: (02%u)03 • N1/B1: 0 (zero, sends and receives a command) • N2/B2: 1 (one, command's ID) • Read Tag: 1200

The **TX** packet is assembled to send a query command according to the structure of a device's protocol, that is, by using `"/h02"` (byte STX 0x02) + `"query"` + `"/h03"` (byte ETX 0x03).

The **RX** packet is assembled with format `"(02"` (starting not including STX 0x02) + `"%u"` + `")03"` (ending not including ETX 0x03), in which the `%u` parameter extracts data in unsigned numerical format.

Usage examples

PROTOCOL	GOAL	CONFIGURATION
TX: \$memory13<CR><LF> RX: \$34.45<CR><LF>	Send a memory command plus a memory position to a device and extract a response's numerical value	<ul style="list-style-type: none"> • ID: 1 (one) • TX: \$memory%u/r/n • RX: (24%f)0D*1 • N1/B1: 1 (one, TX values) • N2/B2: 1 (one, command's ID) • Write Tag: 13 • N1/B1: 0 (zero, sends and receives a command) • N2/B2: 1 (one, command's ID) • Read Tag: 34.45

The **TX** packet requires configuring a memory position. To do so, the **TX** packet is assembled by inserting a variable field `%u`, to be defined by writing a Tag with the *N1/B1* parameter equal to 1 (one, **TX** values). After filling in the variable field `%u` with a numerical value, in this example the value 13, users can send the **TX** packet to a device in the format `"$"` + `"memory"` + `"13"` + `"/r` (byte CR 0x0D) + `"/n` (byte LF 0x0A).

The **RX** packet is assembled with the format "(24" (starting not including "\$") + "%f" + ")0D*1" (ending not including CR 0x0D + discarding of the LF 0x0A byte that follows), in which the %f parameter extracts data in real numerical format.

Usage examples

PROTOCOL	GOAL	CONFIGURATION
TX: #GetData<CR><LF> RX: #12;452;75<CR><LF>	Send a #GetData command to a device and extract three numerical values from the response	<ul style="list-style-type: none"> • ID: 1 (one) • TX: #GetData/r/n • RX: (23%u;%u;%u)0D*1 • N1/B1: 0 (zero, sends and receives a command) • N2/B2: 1 (one, command's ID) • Read BlockTag: <ul style="list-style-type: none"> • Element 1: 12 • Element 2: 452 • Element 3: 75

The **TX** packet is assembled to send a **GetData** command according to the structure of a device's protocol, that is, by using "#" + "GetData" + /r (byte CR 0x0D) + /n (byte LF 0x0A).

The **RX** packet is assembled with format "(23" (starting not including "#") + "%u" + ";" + "%u" + ";" + "%u" + ")0D*1" (ending not including CR 0x0D + discarding of the LF 0x0A byte that follows), in which three %u formats are extracted by using a Block Tag with three Elements, each one representing a %u format.

Usage examples

PROTOCOL	GOAL	CONFIGURATION
TX: <STX>DeviceName<ETX> RX: <STX>Personal Computer<ETX>	Send a DeviceName command to a device and extract a value in String format from the response	<ul style="list-style-type: none"> • ID: 1 (one) • TX: /h02DeviceName/h03 • RX: (02%s)03 • N1/B1: 0 (zero, sends and receives a command) • N2/B2: 1 (one, command's ID) • Read Tag: "Personal Computer"

The **TX** packet is assembled to send a **DeviceName** command according to the structure of a device's protocol, that is, by using /h02 (byte STX 0x02) + "DeviceName" + /h03 (byte ETX 0x03).

The **RX** packet is assembled with format "(02" (starting not including STX 0x02) + "%s" + ")03" (ending not including ETX 0x03), in which the %s parameter extract data in **String** format.

Usage examples

PROTOCOL	GOAL	CONFIGURATION
TX: <STX>DeviceName<ETX> RX: <STX>Personal Computer<ETX>	Send a DeviceName command to a device and extract a substring from the response's value	<ul style="list-style-type: none"> • ID: 1 (one) • TX: /h02DeviceName/h03 • RX: (02%@r#as)03 • N1/B1: 0 (zero, sends and receives a command) • N2/B2: 1 (one, command's ID) • Read Tag: "son"

The **TX** packet is assembled to send a **DeviceName** command according to the structure of a device's protocol, that is, by using /h02 (byte STX 0x02) + "DeviceName" + /h03 (byte ETX 0x03).

The **RX** packet is assembled with format "(02" (starting not including STX 0x02) + "%@r#as" + ")03" (ending not including ETX 0x03), in which the %@r#as parameter extracts a substring between the initial "r" character and the final "a" character from the value "Personal Computer", resulting in a "son" substring.

Tag Reference

All Tags from this Driver must be referenced by their *N/B* parameters, according to the next table.

Available options for Tag parameters

PARAMETER	DESCRIPTION
N1/B1	Function code
N2/B2	Index on the list of commands
N3/B3	Additional parameter, if needed
N4/B4	Additional parameter, if needed

The next table contains all functions available in this Driver.

Available functions for the ASCII Generic Driver

N1/B1	DESCRIPTION	APPLICATION
0	Sends and Receives Commands	Used to perform a full transaction with a device, by sending a TX request packet and receiving an RX response packet
1	TX Values	Used only to define the % variables of a TX packet, without communicating with a device
2	RX Values	Used only to query % variables of an RX packet, without communicating with a device
3	Receives Commands	Only receives an RX response packet, frequently used with devices constantly

N1/B1	DESCRIPTION	APPLICATION
		sending data without the need to receive a TX request packet
4	Sends Commands	Only sends a TX request packet, frequently used with devices not returning an RX response packet
5	Sends a Tag Value	Uses the content of a Tag to represent a TX request packet and sends it to a device
6	Sends and Receives List Commands	Used to trigger a TX request packet that generates the return of multiple RX response packets
7	Receives List Commands	Only receives multiple RX response packets

Sends and Receives Commands

Reading and Writing

N1	0 (zero)
N2	Command's index
N3	Variable's index
N4	Offset of RX message

Used to perform a full transaction with a device, sending a **TX** request packet and receiving an **RX** response packet.

If used on a reading command, returns the variable fields from a response message, one for each Block Element, if applicable, starting with the index of the variable informed in the *N3* parameter. For example, if the **20** command demands a response of type **ABC=%d** and **XYZ=%f** when reading a Block Tag with two Elements in the **0.20.0.0** configuration, the first Element contains the first variable field (decimal) and the second Element contains the second variable field (real number).

Use the *N4* parameter so that the command analyzes the response frame with an offset relative to the first character of the message. For example, to discard the first 100 bytes of a message, configure the *N4* parameter as 100.

TX Values

Reading and Writing

N1	1 (one)
N2	Command's index
N3	Variable's index
N4	0 (zero)

Used only to define the % variables of a **TX** packet, without performing communication with a device. This Tag reads or writes values of **TX**'s % variable fields before sending a command. The data type of each Tag must match the data types informed when defining the command. If a reading command is used, it returns the variable fields of a **TX** message, one for each Block Element, if using a Block Tag, starting with the variable's index indicated in the *N3* parameter.

RX Values

Read-Only

N1	2 (two)
N2	Command's index
N3	Variable's index
N4	0 (zero)

Used only to query % variables of an **RX** packet, without performing communication with a device. This Tag reads values from **RX** packet's % variables update on the last reception of the command informed in the *N2* parameter. The data type of each Tag must match the data types informed when defining the command. It returns the variable fields of an **RX** message, one for each Block Element, if using a Block Tag, starting with the variable's index informed in the *N3* parameter.

Receives Commands

Read-Only

N1	3 (three)
N2	Command's index
N3	Variable's index
N4	Offset of RX message

Only receives an **RX** response packet, normally used with devices that continuously send data, without the need to receive a **TX** request packet. Returns the variable fields from a response message, one for each Block Element, if using a Block Tag, starting with the variable's index informed in the *N3* parameter.

Use the *N4* parameter so that the command analyzes the response frame with an offset relative to the first character of the message. For example, to discard the first 100 bytes of a message, configure the *N4* parameter to 100.

For a better reception of continuous data, users can reduce **IOKit**'s time-out to a very low value, something around 200 ms. In this type of continuous data reception, a time-out is not aimed to monitor a limit for responses to **TX** requests, simply because they are not present in this type of communication.

Sends Commands

Write-Only

N1	4 (four)
N2	Command's index
N3	0 (zero)
N4	Clear response (optional)

Only sends a **TX** request packet, normally used with devices that do not return an **RX** response packet. Indicate in the *N4* parameter a value different from 0 (zero) if this Driver must wait for a response to this command.

Sends a Tag Value

Write-Only

N1	5 (five)
N2	0 (zero)
N3	0 (zero)
N4	Clear response (optional)

Uses the content of this Tag to represent a **TX** request packet and sends it to a device. The message to send is the content of this Tag in **Text** format, usually assembled with functions in an application's script code. Indicate in the *N4* parameter a value different from 0 (zero) if this Driver must wait for a response to this command.

Sends and Receives List Commands

Read-Only

N1	6 (six)
N2	Command's index
N3	Variable's index
N4	Offset of RX message

Used to trigger a **TX** request packet that generates a return of multiple **RX** response packets. This Tag is similar to the command with an **N1 parameter equal to (zero)**, however it generates a list of values for each repeated set of variables found, according to the description of the **RX** command. Users must use this Tag when the number of rows the command returns is unknown, considering that all rows follow the same structure, according to the next example.

```
=>HIS <ENTER>
FEEDER 1 Date: 08/10/99 Time: 08:40:16.740
STATION A
# DATE TIME EVENT LOCAT CURR FREQ GRP SHOT TARGETS
1 02/01/99 08:33:00.365 TRIG $$$$$$ 1 60.00 3 2
2 01/31/99 20:32:58.361 ER $$$$$$ 231 60.00 2 2
3 01/29/99 07:30:11.055 AG T 9.65 2279 60.00 3 2 INST 50
```

Receives List Commands

Read-Only

N1	7 (seven)
N2	Command's index
N3	Variable's index
N4	Offset of RX message

Only receives multiple **RX** response packets. This Tag is similar to the **Receives Commands** Tag, except it returns those lists in the format of the **Sends and Receives List Commands** Tag.

List of ASCII Characters

Codes for ASCII characters

CHAR	DEC	OCT	HEX	CHAR	DEC	OCT	HEX	CHAR	DEC	OCT	HEX	CHAR	DEC	OCT	HEX
(nul)	0	0	0x00	(sp)	32	40	0x20	@	64	100	0x40	`	96	140	0x60
(soh)	1	1	0x01	!	33	41	0x21	A	65	101	0x41	a	97	141	0x61
(stx)	2	2	0x02	"	34	42	0x22	B	66	102	0x42	b	98	142	0x62
(etx)	3	3	0x03	#	35	43	0x23	C	67	103	0x43	c	99	143	0x63
(eot)	4	4	0x04	\$	36	44	0x24	D	68	104	0x44	d	100	144	0x64
(enq)	5	5	0x05	%	37	45	0x25	E	69	105	0x45	e	101	145	0x65
(ack)	6	6	0x06	&	38	46	0x26	F	70	106	0x46	f	102	146	0x66
(bel)	7	7	0x07	'	39	47	0x27	G	71	107	0x47	g	103	147	0x67
(bs)	8	10	0x08	(40	50	0x28	H	72	110	0x48	h	104	150	0x68
(ht)	9	11	0x09)	41	51	0x29	I	73	111	0x49	i	105	151	0x69
(nl)	10	12	0x0a	*	42	52	0x2a	J	74	112	0x4a	j	106	152	0x6a
(vt)	11	13	0x0b	+	43	53	0x2b	K	75	113	0x4b	k	107	153	0x6b
(np)	12	14	0x0c	,	44	54	0x2c	L	76	114	0x4c	l	108	154	0x6c
(cr)	13	15	0x0d	-	45	55	0x2d	M	77	115	0x4d	m	109	155	0x6d
(so)	14	16	0x0e	.	46	56	0x2e	N	78	116	0x4e	n	110	156	0x6e
(si)	15	17	0x0f	/	47	57	0x2f	O	79	117	0x4f	o	111	157	0x6f
(dle)	16	20	0x10	0	48	60	0x30	P	80	120	0x50	p	112	160	0x70
(dc1)	17	21	0x11	1	49	61	0x31	Q	81	121	0x51	q	113	161	0x71
(dc2)	18	22	0x12	2	50	62	0x32	R	82	122	0x52	r	114	162	0x72
(dc3)	19	23	0x13	3	51	63	0x33	S	83	123	0x53	s	115	163	0x73
(dc4)	20	24	0x14	4	52	64	0x34	T	84	124	0x54	t	116	164	0x74
(nak)	21	25	0x15	5	53	65	0x35	U	85	125	0x55	u	117	165	0x75

CHAR	DEC	OCT	HEX	CHAR	DEC	OCT	HEX	CHAR	DEC	OCT	HEX	CHAR	DEC	OCT	HEX
(syn)	22	26	0x16	6	54	66	0x36	V	86	126	0x56	v	118	166	0x76
(etb)	23	27	0x17	7	55	67	0x37	W	87	127	0x57	w	119	167	0x77
(can)	24	30	0x18	8	56	70	0x38	X	88	130	0x58	x	120	170	0x78
(em)	25	31	0x19	9	57	71	0x39	Y	89	131	0x59	y	121	171	0x79
(sub)	26	32	0x1a	:	58	72	0x3a	Z	90	132	0x5a	z	122	172	0x7a
(esc)	27	33	0x1b	;	59	73	0x3b	[91	133	0x5b	{	123	173	0x7b
(fs)	28	34	0x1c	<	60	74	0x3c	\	92	134	0x5c	 	124	174	0x7c
(gs)	29	35	0x1d	=	61	75	0x3d]	93	135	0x5d	}	125	175	0x7d
(rs)	30	36	0x1e	>	62	76	0x3e	^	94	136	0x5e	~	126	176	0x7e
(us)	31	37	0x1f	?	63	77	0x3f	_	95	137	0x5f	(del)	127	177	0x7f

Documentation of I/O Interfaces

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

Configuration of a Driver

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

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

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

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

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

Configuration Dialog Box

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

Setup Tab

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

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

The screenshot shows the 'Setup' tab configuration window. It is divided into several sections:

- Physical Layer:** A dropdown menu set to 'Ethernet'. To the right is a checkbox for 'Start driver OFFLINE' which is unchecked.
- Timeout:** A text input field containing '1000' followed by 'ms'.
- Communication check time:** A text input field containing '5000' followed by 'ms'.
- Connection management:** A sub-section containing:
 - A dropdown menu for 'Mode' set to 'Automatic (managed by the driver)'.
 - A checked checkbox for 'Retry failed connection every' followed by a text input '20' and 'seconds'.
 - An unchecked checkbox for 'Give up after' followed by a text input '1' and 'failed retries'.
 - An unchecked checkbox for 'Disconnect if non-responsive for' followed by a text input '0' and 'seconds'.
- Logging Options:** A sub-section containing:
 - An unchecked checkbox for 'Log to File:' followed by a text input 'C:\eeLogs\MicrolokII_%DATE%.log'.
 - A text input for 'File size limit (MB):' containing '0' with '(0 is unlimited)' in parentheses.

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

OPTION	DESCRIPTION
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	<p>Enable this option and configure the name of a file to write a log. Log files can be large, so use this option for short periods of time, only for testing and debugging purposes. If the %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 OFDAh. 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</p>
File size limit (MB)	<p>Configure the log file size limit, in megabytes. A value equal to 0 (zero) means that there is no size limit for the log file</p>

Serial Tab

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

Serial

Port:

Baud rate:

Data bits:

Parity:

Stop bits:

Enable 'ECHO' suppression

Handshaking

DTR control:

RTS control:

Wait for CTS before send

CTS timeout: ms

Delay before send: ms

Delay after send: ms

Inter-byte delay (microseconds): μ s

Inter-frame delay (milliseconds): ms

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 COMn , 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' suppression	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

Transport: TCP/IP ▼

PING before connecting
 Timeout: 4000 ms
 Retries: 1

Listen for connections on port: 0

Share listen port with other processes

Interface: (All Interfaces) ▼

Use IPv6 Use SSL SSL Settings

Enable 'ECHO' supression

IP Filter:

Connect to

<input type="checkbox"/> Main IP:	 	Port:	502	<input type="checkbox"/> Local port:	0
<input type="checkbox"/> Backup IP 1:	 	Port:	0	<input type="checkbox"/> Local port:	0
<input type="checkbox"/> Backup IP 2:	 	Port:	0	<input type="checkbox"/> Local port:	0
<input type="checkbox"/> Backup IP 3:	 	Port:	0	<input type="checkbox"/> Local port:	0

Ethernet tab

Available options on the Ethernet tab

OPTION	DESCRIPTION
Transport	Select the value TCP/IP for a TCP socket (<i>stream</i>) or select the value UDP/IP to use a UDP socket (<i>connectionless datagram</i>)
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' supression	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 (<i>Firewall</i>). 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
	<p>of determining a successful connection before trying to open a socket with a device. The time-out of a connection with a socket can be very high. The available options are:</p> <ul style="list-style-type: none"> • 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

Select the modem to use:

▼ Modem settings...

Dial Number:

Accept incoming calls

Modem tab

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

Available options on the Modem tab

OPTION	DESCRIPTION
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

AT command:

Connection timeout: 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" 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 value **Manual**, then the physical layer remains in this status until forced to connect
- **2:** Physical layer connected, that is, the physical layer is ready for use. This **DOES NOT** mean a device is connected, only that the access layer is working

IO.SetConfigurationParameters

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

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

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

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

IMPORTANT

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

Properties

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

IO.ConnectionMode

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

IO.GiveUpEnable

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

IO.GiveUpTries

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

IO.InactivityEnable

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

IO.InactivityPeriodSec

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

IO.RecoverEnable

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

IO.RecoverPeriodSec

9 Delay time between two connection attempts, in seconds.

NOTE

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

IO.StartOffline

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

NOTE

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

IO.TimeoutMs

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

IO.Type

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, 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:0877:0000:0000:0000:0000:*)**: Accepts connections from IPv6 addresses in the interval between fe80:03bf:0877:0000:0000:0000:0000:0000 and fe80:03bf:0877:0000:0000:0000:ffff:ffff
- **192.168.0.10, 192.168.0.15, 192.168.0.20**: Accepts connections from IPv4 addresses 192.168.0.10, 192.168.0.15, and 192.168.0.20
- **~192.168.0.95, 192.168.0.***: Accepts connections from IPv4 addresses in the interval between 192.168.0.0 and 192.168.0.255, except the IPv4 address 192.168.0.95

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

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

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

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

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

IO.Ethernet.ListenIP

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

IO.Ethernet.ListenPort

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

IO.Ethernet.MainIP

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

IO.Ethernet.MainLocalPort

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

IO.Ethernet.MainLocalPortEnable

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

IO.Ethernet.MainPort

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

IO.Ethernet.PingEnable

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

IO.Ethernet.PingTimeoutMs

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

IO.Ethernet.PingTries

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

IO.Ethernet.ShareListenPort

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

IO.Ethernet.SupressEcho

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

IO.Ethernet.Transport

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

IO.Ethernet.UseIPv6

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

Modem Interface Configuration

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

I/O Tags

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

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

IMPORTANT

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

IO.TAPI.ConnectionBaudRate

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 COMn port.

IO.Serial.RTS

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

IO.Serial.StopBits

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

IO.Serial.SuppressEcho

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

IO.Serial.WaitCTS

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

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
2.0.5	06/13/2025	M. Ludwig	<ul style="list-style-type: none"> Driver updated to IOKit library version 3.0 and Visual Studio 2022 (<i>Case 37956</i>).
2.0.4	09/13/2023	C. Mello	<ul style="list-style-type: none"> Adjustments to preserve the last valid data received in a continuous data reading by using a Tag with the <i>N1/B1</i> parameter equal to 3 (three, Receives Commands) (<i>Case 34817</i>).
2.0.3	08/29/2019	C. Mello	<ul style="list-style-type: none"> Platform update on this Driver's source code (<i>Case 27445</i>).
2.0.2	05/22/2018	C. Mello	<ul style="list-style-type: none"> Added a parameter to discard <i>n</i> bytes after finding the data reception's

VERSION	DATE	AUTHOR	COMMENTS
			terminator byte (<i>Case 24265</i>).
2.0.1	12/04/2017	C. Mello	<ul style="list-style-type: none"> Adjustments with the @ symbol to extract data in the %s format without including the initialization character (<i>Case 15340</i>). Added a function to determine the starting byte and the ending byte of data reception (<i>Case 17078</i>). Driver updated and standardized for the new standard of IOKit library version 2.0 (<i>Case 23539</i>).
1.11.1	04/06/2011	C. Mello	<ul style="list-style-type: none"> Added support for interpreting decimal values in BCD format for %n data type (<i>Case 12162</i>). Added support for WinCE platform (<i>Case 10917</i>).
1.10.1	05/11/2010	C. Mello	<ul style="list-style-type: none"> Fixed a problem when receiving data when the RX message starts with an "/hxx" value (<i>Case 11377</i>).
1.9.1	09/16/2009	M. Ludwig	<ul style="list-style-type: none"> Driver updated to IOKit library version 1.15. Fixed an error when writing Block Tags (<i>Case 10725</i>).
1.8.1	05/13/2009	C. Mello	<ul style="list-style-type: none"> Adjustments in the command with the <i>N1</i> parameter equal to 7 (seven) to receive data as a list (<i>Case 10472</i>).
1.7.1	06/05/2008	C. Mello	<ul style="list-style-type: none"> Adjustments to ignore the delimiter character as part of data (<i>Case 9566</i>). Adjustments in the command with the <i>N1</i> parameter equal to 5 (five) to dismiss consistencies of the <i>N2</i> parameter.
1.6.1	06/01/2007	C. Mello	<ul style="list-style-type: none"> Adjustments to prevent duplicating this Driver's configuration properties after restarting at run time (<i>Case 8031</i>).
1.5.1	01/23/2007	C. Mello	<ul style="list-style-type: none"> Added the /hxx format to inform special characters in TX and RX fields of this

VERSION	DATE	AUTHOR	COMMENTS
			<p>Driver's extra parameters (Case 7526).</p> <ul style="list-style-type: none"> Adjustments in the <i>N4/B4</i> parameter to enable receiving data via commands with the <i>N1/B1</i> parameter equal to 4 (four) and 5 (five).
1.4.1	12/22/2005	C. Mello	<ul style="list-style-type: none"> Defined a %b data type. Adjustments on this Driver's extra parameters dialog box. General revision of all source code.
		M. Salvador	<ul style="list-style-type: none"> Added support for unordered lists of messages.
1.3.1	07/20/2005	M. Salvador	<ul style="list-style-type: none"> Configured the maximum number of bytes and Block Elements. Defined a %t data type. Defined the @ character at the beginning of Strings, and also defined characters in hexadecimal (hXX), both at the beginning and at the end of Strings. Defined new commands with the <i>N1/B1</i> parameters equal to 6 (six) and 7 (seven). Added an offset on the RX response frame with the <i>N4</i> parameter. Fixed an error in which the %f data type did not interpreted signs. Added a list of ASCII characters on the documentation.
1.2.1	07/12/2005	C. Mello	<ul style="list-style-type: none"> Defined a %d data type as a signed decimal value. Defined a %u data type as an unsigned decimal value.
		M. Salvador	<ul style="list-style-type: none"> Implemented a command only to receive data with the <i>N1/B1</i> parameter equal to 3 (three).

VERSION	DATE	AUTHOR	COMMENTS
			<ul style="list-style-type: none">• Implemented a command only to send data with the <i>N1/B1</i> parameter equal to 4 (four).• Implemented a command only to send data from Tag's content in Text format with the <i>N1/B1</i> parameter equal to 5 (five).
1.1.1	09/14/2004	M. Salvador	<ul style="list-style-type: none">• Added support for hexadecimal values, <i>/r</i>, and <i>/n</i>.
1.0.1	09/08/2004	M. Salvador	<ul style="list-style-type: none">• Original version of this Driver.

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.

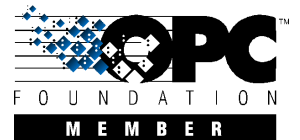
www.elipse.com.br

kb.elipse.com.br

forum.elipse.com.br

www.youtube.com/elipsesoftware

elipse@elipse.com.br



Gartner, 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)