Altus AlnetIv2 Driver

File Name	AlnetIv2.dll
Manufacturer	Altus Sistemas de Automação S.A.
Devices	AL-2000, AL-3000, AL-600, QUARK, PICCOLO, and PONTO
Protocol	Alnet I version 2.0
Version	2.4.10
Last Update	09/01/2025
Platform	Win32
Dependencies	IOKit version 1.6 or later
Superblocks	Yes
Level	0

Introduction

This is Altus AlnetIv2 Driver for communication between **Elipse Software** systems and Altus controllers from AL-2000, AL-3000, AL-600, QUARK, PICCOLO, and PONTO series, using the Alnet I version 2.0 protocol.

This Driver is the result of unifying the old **AL2000**, **AL2000MNS**, **AL2000-Modem**, and **AL2000 Timestamp** Drivers. There has been an effort to keep compatibility with these versions regarding the configuration of Tags. Except for the Tag with the *B2* parameter equal to 6000, **Reading Data Blocks with Timestamp**, which was originally identified by the *B2* parameter equal to 2000, all other Tags follow the same standards for previous Drivers and work without changes in existing applications for these versions.

Although this Driver was designed to allow communication on Alnet-I networks, it also allows communication on Alnet-II networks by using a manufacturer's library, which performs a conversion between these two protocols.

NOTE

This Driver supports Superblocks or Tag grouping, that is, the **EnableReadGrouping** property.

Preparing a Device

Communication can be performed directly using an RS232 cable or using an AL1413, via RS485 cable, for longer distances. Configuring a device must be performed before using manufacturer's MasterTool program.

This Driver can also communicate with PLCs on Alnet-II networks, if an AL2400/S gateway is used. In this case, users must specify not only the PLC address on I/O Tags but also the sub-network address multiplied by 100 to the *N1* or *B1* parameter of each Tag. For more information, please check topics **[B] Parameters for Block Tag Addressing** and **[N] Parameters for PLC Tag Addressing**. The default configuration of PLC's serial communication, configurable on some devices, and for this Driver is the following:

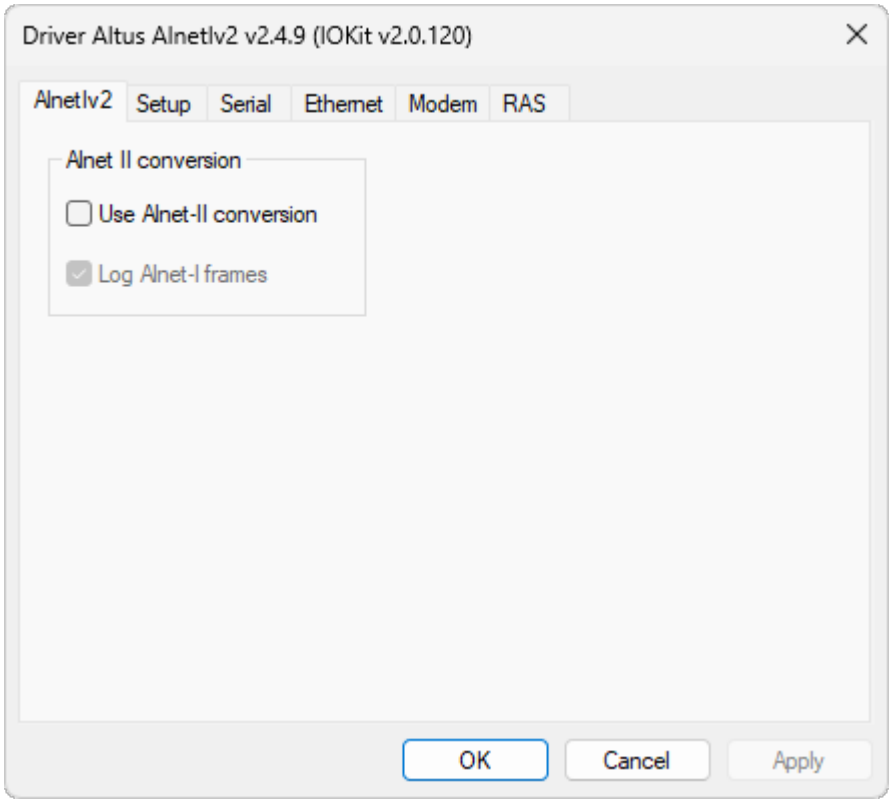
- **Speed**: 9600 bps
- **Number of data bits per character**: 8 (eight)
- **Parity**: even
- **Number of start bits**: 1 (one)

- **Number of stop bits**: 1 (one)

Driver Configuration

This Driver does not use **[P]** parameters. All configuration regarding communication must be performed on this Driver's configuration window. On this window there is a specific tab, **AlnetIv2**, and the **Setup**, **Serial**, **Ethernet**, **Modem**, and **RAS** tabs, specific for **Elipse Software**'s **IOKit** library, used by this Driver. For more information about that library, please check topic **Documentation of I/O Interfaces**.

Configuring Properties



AlnetIv2 tab

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

Available option on the AlnetIv2 tab

OPTION	DESCRIPTION
Use Alnet-II conversion	Convert Alnet-I network packets to Alnet-II network packets, allowing communication in these networks
Log Alnet-I frames	If the **Use Alnet-II conversion** option is enabled, allows frames from Alnet-I network packets to be viewed in logs before passing through the layer to convert Alnet-I network packets into Alnet-II network packets

Tag Reference

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

Introduction

This Driver's Tags allow reading from and writing to a PLC's operands and operand tables. Users can also access operands from this Driver's internal memory, changed by unsolicited messages received from devices, by setting 9999 to the *N1* or *B1* parameter in Tags that support unsolicited messages. This Driver, following the standard implemented earlier in the old **AL2000 Timestamp** Driver, allows access by unsolicited messages only the **Memory**, **Memory Table**, **Octet**, and **Auxiliary** operands. Writing unsolicited messages using Tags is not allowed. Such values can only be changed by messages from PLCs. When reading values from unsolicited messages, the **Timestamp** property of a Tag indicates the moment of the last change of value or receiving a message. For more information about receiving unsolicited messages from a PLC, please check topic **Unsolicited Messages (MNS)**.

Tags for reading and writing operands can be configured using *N* and *B* parameters as in previous versions, or alternatively, if this Driver is used with **Elipse E3**, **Elipse Power**, or **Elipse Water**, by using **Strings**. Configuring by **Strings** follow the same standard used in OPC Drivers, by using the **Device** and **Item** properties of Tags in **Elipse E3**, **Elipse Power**, or **Elipse Water**. For more information, please check topic **Configuration via Strings**.

[N] Parameters for PLC Tag Addressing

N1	Network address × 100 + PLC number. Please check the next **note** for more information
N2	Data type. Please check table **Supported data types for PLC Tags** for more information. Add 10000 for unsigned integers, that is, **Memory** and **Memory Table** data types
N3	Address. Please check the next **note** for more information
N4	Bit or table position

NOTE

Some addresses are expressed in octal in a PLC. To correctly read those values in **Elipse Software** applications, their addresses must be converted to decimal.

The next table shows a summary of data types that can be read by these Tags and their configuration, which are described on the next topics.

Supported data types for PLC Tags

DATA TYPE	MNEMONIC	N2	N3	N4	MODE
Memory	M<add>	0 (zero)	Address	**0**: Value, **1-8**: Bit	Reading and writing
Memory Table	TM<add>:<pos>	1 (one)	Address	Position	Reading and writing
Memory Bit	bit <n> of M<add>	2 (two)	Address	*n*	Reading and writing
Decimal	D<add>	3 (three)	Address	**0**: Value, **1-8**: Bit	Reading and writing
Decimal Table	TD<add>:<pos>	4 (four)	Address	Position	Reading and writing
Auxiliary	A<add>	5 (five)	Address	**0**: Value, **1-8**: Bit	Reading and writing

DATA TYPE	MNEMONIC	N2	N3	N4	MODE
Input and Output Octet	E<add> and S<add>	6 (six)	Address	**0**: Value, **1-8**: Bit	Reading and writing
32-Bit Integer	I<add>	7 (seven)	Address	**0**: Value, **1-8**: Bit	Reading and writing
Bit in Octet	E<add > and S<add>	8 (eight)	Address	Bit (0-7)	Write-only
Bit in Auxiliary	A<add>	9 (nine)	Address	Bit (0-7)	Write-only
Real or Float	F<add>	10	Address	**0**: Value, **1-8**: Bit	Reading and writing
Real or Float Table	TF<add>:<pos>	11	Address	Position	Reading and writing
32-Bit Integer Table	TI<add>:<pos>	12	Address	Position	Reading and writing
Reading Data with Timestamp		13	Address of decimal table	Auxiliary address	Read-only
Release all forced nodes	-	200	-	-	Write-only

Memory

Reading and Writing

N1	PLC address, in the format Network number × 100 + PLC number or 9999 for MNS
N2	0 (zero)
N3	Address
N4	Bit. Possible values are **0**: Reads the value of the integer operand or greater than 0 (zero) to read only the specified bit. The least significant bit is 1 (one)

This Tag reads and writes **Memory**-type operands.

Memory Table

Reading and Writing

N1	PLC address, in the format Network number × 100 + PLC number or 9999 for MNS
N2	1 (one)
N3	Table
N4	Position

This Tag reads and writes **Memory Table**-type operands.

Memory Bit

Reading and Writing

N1	PLC address, in the format Network number × 100 + PLC number or 9999 for MNS
N2	2 (two)
N3	Address
N4	Bit to access. The least significant bit is 0 (zero)

This Tag reads and writes bits of **Memory Bit**-type operands.

NOTE

This Tag was kept for compatibility and users must avoid using it in new applications. These applications must use the Tag to access **Memory**-type operands with the *N4* parameter greater than 0 (zero).

Decimal

Reading and Writing

N1	PLC address, in the format Network number × 100 + PLC number
N2	3 (three)
N3	Address
N4	Bit. Possible values are **0**: Reads the value of the integer operand or greater than 0 (zero) to read only the specified bit. The least significant bit is 1 (one)

This Tag reads and writes **Decimal**-type operands.

Decimal Table

Reading and Writing

N1	PLC address, in the format Network number × 100 + PLC number
N2	4 (four)
N3	Table
N4	Position

This Tag reads and writes decimal tables.

Auxiliary

Reading and Writing

N1	PLC address, in the format Network number × 100 + PLC number or 9999 for MNS
N2	5 (five)
N3	Address
N4	Bit. Possible values are **0**: Reads the value of the integer operand or greater than 0 (zero) to read only the specified bit. The least significant bit is 1 (one)

This Tag reads and writes **Auxiliary**-type operands.

Input and Output Octet

Reading and Writing

N1	PLC address, in the format Network number × 100 + PLC number or 9999 for MNS
N2	6 (six)
N3	Address
N4	Bit. Possible values are **0**: Reads the value of the integer operand or greater than 0 (zero) to read only the specified bit. The least significant bit is 1 (one)

This Tag reads and writes in digital input and output octets, respectively.

32-Bit Integer

Reading and Writing

N1	PLC address, in the format Network number × 100 + PLC number
N2	7 (seven)
N3	Address
N4	Bit. Possible values are **0**: Reads the value of the integer operand or greater than 0 (zero) to read only the specified bit. The least significant bit is 1 (one)

This Tag reads and writes **32-Bit Integer**-type operands.

Bit in Octet

Read-Only

N1	PLC address, in the format Network number × 100 + PLC number
N2	8 (eight)
N3	Address
N4	Bit

This Tag writes to octet bits.

NOTE

This Tag was kept for compatibility and users must avoid using it in new applications. These applications must use the Tag to access **Input and Output Octet**-type operands with the *N4* parameter equal to 0 (zero).

Bit in Auxiliary

Write-Only

N1	PLC address, in the format Network number × 100 + PLC number
N2	9 (nine)
N3	Address
N4	Bit

This Tag reads and writes bits of **Auxiliary**-type operands.

NOTE

This Tag was kept for compatibility and users must avoid using it in new applications. These applications must use the Tag to access **Auxiliary**-type operands with the *N4* parameter greater than 0 (zero).

Real or Float

Reading and Writing

N1	PLC address, in the format Network number × 100 + PLC number
N2	10
N3	Address
N4	Bit. Possible values are **0**: Reads the value of the integer operand or greater than 0 (zero) to read only the specified bit. The least significant bit is 1 (one)

This Tag reads and writes **Real**- or **Float**-type operands.

NOTE

For **Float** values with non-normalized numbers, this Driver assumes the value 0 (zero) for underflow cases and **null** for overflow cases.

Real or Float Table

Reading and Writing

N1	PLC address, in the format Network number × 100 + PLC number
N2	11
N3	Table
N4	Position

This Tag reads and writes **Real** or **Float** tables.

NOTE

For **Float** values with non-normalized numbers, this Driver assumes the value 0 (zero) for underflow cases and **null** for overflow cases.

32-Bit Integer Table

Reading and Writing

N1	PLC address, in the format Network number × 100 + PLC number
N2	12
N3	Table
N4	Position

This Tag reads and writes **32-Bit Integer** tables.

Reading Data with Timestamp

Read-Only

N1	PLC address, in the format Network number × 100 + PLC number
N2	13
N3	Address of a table for decimal reading
N4	Auxiliary address

This Tag reads variables from decimal tables with timestamp, with constant intervals between variables. For more information about the usage of this resource, please check topic **Reading Data with Timestamp**.

[B] Parameters for Block Tag Addressing

The next table contains data types that can be read with these Tags and their respective configuration, which are described on the next topics.

Supported data types for Block Tags

DATA TYPE	B2	B3	B4	MODE
Memory Table	Table number	Table's initial position	-	Reading and writing
Decimal Table	1000 + table number	Table's initial position	-	Reading and writing
Input and Output Bus Status	2000	Bus number	-	Read-only
Communication Status	2001	Reset of variables	-	Read-only
Device Status	2002	-	-	Read-only
Memory Block	3000	Initial address	-	Reading and writing
Real or Float Table	4000 + table number	Table's initial position	-	Reading and writing

DATA TYPE	B2	B3	B4	MODE
32-Bit Integer Table	6000 + table number	Table's initial position	-	Reading and writing
Real or Float Block	5000	Initial address	-	Reading and writing
32-Bit Integer Block	5003	Initial address	-	Reading and writing
Data Block with Timestamp	5005	Decimal table's initial position	Auxiliary address	Write-only

Memory Table

Reading and Writing

B1	PLC address, in the format Network number × 100 + PLC number or 9999 for MNS
B2	Table number
B3	Position
B4	Not used

This Tag reads and writes memory tables.

Decimal Table

Reading and Writing

B1	PLC address, in the format Network number × 100 + PLC number
B2	1000 + Table number
B3	Position
B4	Not used

This Tag reads and writes decimal tables.

Input and Output Bus Status

Read-Only

B1	PLC address, in the format Network number × 100 + PLC number
B2	2000
B3	Bus number
B4	Not used

This Tag reads the status of an input and output bus and its modules. The meaning of the values of its returned Block Elements are described on the next table.

Input and output bus Block Elements

ELEMENT	DESCRIPTION	COMMENT
1	TOT_BAR	Total number of possible input and output buses on a PLC
2	TIPO_BAR	Bus type. Possible values are **0**: AL-1000 bus, **1**: AL-3000 bus, or **2**: PONTO bus
3	ST_BAR	Bus status
4	NUM_MOD	Number of possible input and output modules on the selected bus
5	TIPO_MOD_0	Type of module 0 (zero)
6	ST_MOD_0	Status of module 0 (zero)
7	TIPO_MOD_1	Type of module 1 (one)
8	ST_MOD_1	Status of module 1 (one)
...	...	...
	TIPO_MOD_N	Type of module *n*
	ST_MOD_N	Status of module *n*

Communication Status

Read-Only

B1	PLC address, in the format Network number × 100 + PLC number
B2	2001
B3	Reset of variables. Possible values are **0**: Data values about reception and transmission are not changed or **1**: Data values about reception and transmission are zeroed
B4	Not used

This Tag reads information about the status of ALNET II communication. The meaning of the values of its Block Elements are described on the next table.

Block Elements of communication status

ELEMENT	DESCRIPTION	COMMENT
1	DADOS_TX	Number of transmissions without errors
2		Number of transmissions with collision errors
3		Number of transmissions with underrun errors
4		Number of transmissions without hardware's ACK reception
5		Number of canceled transmissions after exhausting all retries
6		Number of service time-outs
7		Number of transmission buffers faults
8		Reserved
9		Reserved
10	DADOS_RX	Number of receptions without errors
11		Number of receptions with collision errors
12		Number of receptions with overrun errors
13		Number of receptions with CRC errors
14		Number of receptions with alignment errors
15		Number of receptions of packets with wrong sizes
16		Number of packet time-outs
17		Number of reception buffers faults
18		Reserved
19	DADOS_COM	Communication speed
20		Node address
21		Local sub-net address
22		Multicast groups containing this node
23		Reserved
24		Reserved
25		Intrabus time-out
26		Interbus time-out
27		Packet time-out

ELEMENT	DESCRIPTION	COMMENT
28		Maximum number of transmission retries
29		Identifier of the node on the network, with 20 bytes
30	DADOS_CNX	Type of physical connection. Possible values are **0**: Electrical default or **1**: Optical
31		Redundancy of physical connections. Possible values are **0**: Without redundancy or **1**: With redundancy
32		Period for redundancy test, in seconds
33		Time for commutation of physical connection, in seconds
34		Reserved
35		Reserved
36		Selected physical connection. Possible values are 1 (one) or 2 (two)
37		Status of connection 1 (one). Possible values are **0**: OK or **1**: On failure
38		Status of connection 2 (two). Possible values are **0**: OK or **1**: On failure
39		Reserved
40		Forced physical connection. Possible values are 0 (Not forced), 1 (one), or 2 (two)
41		Forced status of connection 1 (one). Possible values are **0**: OK or **1**: On failure
42		Forced status of connection 2 (two). Possible values are **0**: OK or **1**: On failure

Device Status

Read-Only

B1	PLC address, in the format Network number × 100 + PLC number
B2	2002
B3	Not used
B4	Not used

This Block Tag reads information about the status of a device. The meaning of the values of its Block Elements are described on the next table.

Block Elements of a device status

ELEMENT	MNEMONIC	DESCRIPTION
1	MODELO_UCP	CPU model
2	VERS_EXE_H	Version of executable
3	VERS_EXE_L	
4	MODO_OP_1	Mode of operation 1 (one)
5	COD_MESG_1	Code of message 1 (one)
6	RAM_LIV_2H	Free RAM space of application bank 2 (two)
7	RAM_LIV_2H	
8	RAM_LIV_1H	Bank 1 (one)
9	RAM_LIV_1L	
10	STAT_RAM_1	RAM status of application 1 (one)
11	TC_INST_H	Time of snapshot cycle
12	TC_INST_L	
13	TC_MED_H	Time of average cycle
14	TC_MED_L	
15	TC_MAX_H	Time of maximum cycle
16	TC_MAX_L	
17	TC_MIN_H	Time of minimum cycle
18	TC_MIN_L	
19	PER_E018	Period of E018 module call
20	PER_E019	Period of E019 module call
21	-	-
22	TEMPO_PROG	Maximum time for application execution
23	STAT_RAM_2	RAM status of application 2 (two)
24	MODO_OP_2	Mode of operation 2 (two)
25	RAM_LIV_8H	Free RAM space of application bank 8 (eight)
26	RAM_LIV_8L	
27	RAM_LIV_7H	Bank 7 (seven)
28	RAM_LIV_7L	
29	RAM_LIV_6H	Bank 6 (six)
30	DA RAM_LIV_6L	
31	RAM_LIV_5H	Bank 5 (five)

ELEMENT	MNEMONIC	DESCRIPTION
32	RAM_LIV_5L	
33	RAM_LIV_4H	Bank 4 (four)
34	RAM_LIV_4L	
35	RAM_LIV_3H	Bank 3 (three)
36	RAM_LIV_3L	
37	STATUS_EPR	Status of an application's EPROM
38	EPR_LIV_8H	Free EPROM space of application bank 8 (eight)
39	EPR_LIV_8L	
40	EPR_LIV_7H	Bank 7 (seven)
41	EPR_LIV_7L	
42	EPR_LIV_6H	Bank 6 (six)
43	EPR_LIV_6L	
44	EPR_LIV_5H	Bank 5 (five)
45	EPR_LIV_5L	
46	EPR_LIV_4H	Bank 4 (four)
47	EPR_LIV_4L	
48	EPR_LIV_3H	Bank 3 (three)
49	EPR_LIV_3L	
50	EPR_LIV_2H	Bank 2 (two)
51	EPR_LIV_2L	
52	EPR_LIV_1H	Bank 1 (one)
53	EPR_LIV_1L	
54	COD_MESG_2	Code of message 2 (two)
55	COD_MESG_3	Code of message 3 (three)
56	COD_MESG_4	Code of message 4 (four)
57	IDENT_0	Secondary product identification (ASCII)
58	IDENT_1	
59	IDENT_2	
60	IDENT_3	
61	IDENT_4	
62	IDENT_5	
63	IDENT_6	
64	IDENT_7	
65	EPR_LIV_16H	Bank 16
66	EPR_LIV_16L	

ELEMENT	MNEMONIC	DESCRIPTION
67	EPR_LIV_15H	Bank 15
68	EPR_LIV_15L	
69	EPR_LIV_14H	Bank 14
70	EPR_LIV_14L	
71	EPR_LIV_13H	bank 13
72	EPR_LIV_13L	
73	EPR_LIV_12H	Bank 12
74	EPR_LIV_12L	
75	EPR_LIV_11H	Bank 11
76	EPR_LIV_11L	
77	EPR_LIV_10H	Bank 10
78	EPR_LIV_10L	
79	EPR_LIV_9H	Bank 9 (nine)
80	EPR_LIV_9L	

Memory Block

Reading and Writing

B1	PLC address, in the format Network number × 100 + PLC number
B2	3000
B3	Initial address
B4	Not used

This Tag reads and writes blocks of **Memory**-type operands.

Real or Float Table

Reading and Writing

B1	PLC address
B2	4000 + Table number
B3	Position
B4	Not used

This Tag reads and writes tables of **Real**- or **Float**-type operands.

NOTE

For **Float** values with non-normalized numbers, this Driver assumes the value 0 (zero) for underflow cases and **null** for overflow cases.

32-Bit Integer Table

Reading and Writing

B1	PLC address
B2	6000 + Table number
B3	Position
B4	Not used

This Tag reads and writes tables of **32-Bit Integer**-type operands.

Real or Float Block

Reading and Writing

B1	PLC address, in the format Network number × 100 + PLC number
B2	5000
B3	Initial address
B4	Not used

This Tag reads and writes blocks of **Real**- or **Float**-type operands.

NOTE

For **Float** values with non-normalized numbers, this Driver assumes the value 0 (zero) for underflow cases and **null** for overflow cases.

32-Bit Integer Block

Reading and Writing

B1	PLC address, in the format Network number × 100 + PLC number
B2	5003
B3	Initial address
B4	Not used

This Tag reads and writes blocks of **32-Bit Integer**-type operands.

Data Block with Timestamp

Reading and Writing

B1	PLC address, in the format Network number × 100 + PLC number
B2	5005
B3	Initial position of a decimal table
B4	Auxiliary address

This Tag allows performing a reading on decimal tables with timestamp. For more information about the usage of this resource, please check topic **Reading Data with Timestamp**.

Configuration via Strings

This Driver allows configuring Tags via **Strings**, using the same formats used on OPC Drivers. Configuring via **Strings** is performed using the **Device** and **Item** properties of **Elipse E3**, **Elipse Power**, or **Elipse Water** Tags.

NOTE

Configuring via **Strings** was not available for **Elipse SCADA** by the time this version was released.

The **Device** property can be used to define a PLC address, using the **<net>.plc:** format, in which **net** is an optional parameter specifying a network number, and **plc** specifies a PLC address. If the **Device** property is left empty, the address must be informed in the **Item** property using the same format, followed by a colon. If the network address is not present, consider it as 0 (zero). If the address is defined in the **Device** property, then the final colon can be dismissed. The **Item** property defines operand's data to access. The next table contains mnemonics that can be used by this Driver.

Supported mnemonics

MNEMONIC	DESCRIPTION	MODE
<<net>.plc:>Ax	Auxiliary	Reading and writing
<<net>.plc:>Ax.b	Auxiliary bit	Reading and writing
<<net>.plc:>Dx	Decimal	Reading and writing

MNEMONIC	DESCRIPTION	MODE
<<net>.plc:>Dx.b	Decimal bit	Reading and writing
<<net>.plc:>Ex	Input	Read-only
<<net>.plc:>Ex.b	Input bit	Read-only
<<net>.plc:>Fx	Real (Float)	Reading and writing
<<net>.plc:>Ix	Integer (32-bits)	Reading and writing
<<net>.plc:>Mx	Memory	Reading and writing
<<net>.plc:>Mx.b	Memory bit	Reading and writing
<<net>.plc:>Sx	Output	Write-only
<<net>.plc:>Sx.b	Output bit	Write-only
<<net>.plc:>TDx.y	Decimal table	Reading and writing
<<net>.plc:>TFx.y	Real table (Float)	Reading and writing
<<net>.plc:>TIx.y	Integer table (32-bits)	Reading and writing
<<net>.plc:>TMx.y	Memory table	Reading and writing
<<net>.plc:>BSx	Input and output bus status	Read-only
<<net>.plc:>CSx	Communication status	Read-only
<<net>.plc:>ES	Device status	Read-only

In which:

- Bold characters represent fixed mnemonics identifying commands, while regular characters represent values that must be replaced
- The *y* character represents the initial position on a table
- The *b* character represents a bit number, ranging from 0 (zero, the least significant bit) to 31 (the most significant bit), but the actual maximum number depends on the number of bits on a data type

The next table presents usage examples of mnemonics and their respective conversion into *N* or *B* parameters, according to the internal conversion performed by this Driver.

Usage examples of mnemonics with PLC-type Tags

DEVICE	ITEM	N1 OR B1	N2 OR B2	N3 OR B3	N4 OR B4
	1:D8	1 (one)	3 (three)	8 (eight)	0 (zero)
	2.5:F50	205	10	50	0 (zero)
8.2:	E7	802	6 (six)	7 (seven)	0 (zero)
	120:S5	120	6 (six)	5 (five)	0 (zero)
5.4:	A120	504	5 (five)	120	0 (zero)
	2:M8	2 (two)	0 (zero)	8 (eight)	0 (zero)
	6.5:TD6.123	605	4 (four)	6 (six)	123
2.3	TF84.5	203	11	84	5 (five)

DEVICE	ITEM	N1 OR B1	N2 OR B2	N3 OR B3	N4 OR B4
6	TM5.64	6 (six)	1 (one)	5 (five)	64
7.1:	M125.1	701	0 (zero)	125	2 (two)

NOTE

E and **S** mnemonics are both mapped to PLC Tags with the *N2* parameter equal to 6 (six). The difference is only on the operation itself, because only readings in **E** operands and writings in **S** operands are allowed.

Reading Data with Timestamp

This Driver allows reading data with timestamp retrieved from a PLC indicating the moment of acquisition.

Acquisition modes are by **reading data blocks simultaneously acquired**, that is, with the same timestamp, and by **reading data blocks acquired at constant intervals**. In both cases, the controller software must be capable of satisfying certain requirements. These modes are detailed on the next topics.

Data Blocks Acquired Simultaneously

In this mode, users can read variables from decimal tables with timestamps when there is a set of variables that were stored simultaneously. Thus, every set of variables with its linked time information forms a structure, which is repeated several times, hence forming a table. Every structure must be in the format described next.

```
Data 1 - HHMMSS (Example: 19:02 and 34 seconds are stored as 190234)
Data 2 - DDMMYY (Example: February 18th, 1998 is stored as 180298)
Data 3 - value 1
Data 4 - value 2
(...)
Data N - value N - 2
```

The maximum allowed size for a table is 255 integer operands, or 1020 bytes.

For example, on a table with 255 positions, users want to store several readings of 5 (five) analog variables with timestamps. Thus, that table after filled has the configuration described on the next table, assuming the current date is February 18th, 1998, 09:00:00, with acquisitions at every minute.

Example of a data structure with timestamps

TABLE POSITION	VALUE	TABLE POSITION	VALUE	TABLE POSITION	VALUE
0	090000	**7**	090100	**14**	090200
1	180298	**8**	180298	**15**	180298
2	Data	**9**	Data	**16**	Data
3	Data	**10**	Data	**17**	Data
4	Data	**11**	Data	**18**	Data
5	Data	**12**	Data	**19**	Data
6	Data	**13**	Data	**20**	Data

And so on, until the table is completely filled.

NOTE

Since there are only 2 (two) digits for the year, this Driver is only prepared to work up to 2098.

To control access to a table, there are 4 (four) bits of an auxiliary memory in a PLC:

- **BIT 0 (ask_reading_permission)**: Performs a request to access the table to a PLC
- **BIT 1 (reading_allowed)**: Checks if the request was granted
- **BIT 2 (clear)**: This Driver writes 1 (one) to this bit and the PLC is then responsible for zeroing the table and turn this bit back to 0 (zero)
- **BIT 3 (table_zeroed)**: This bit indicates whether that table is zeroed

To read stored data, users must use a Block Tag with the *B2* parameter equal to 6000, where the *B3* parameter indicates the initial position of the table and the *B4* parameter indicates the address of the byte containing the auxiliary bits described earlier.

When reading, if there is more than one set of values or structures on that table, this Driver returns a list of values for an **Elipse Software** application, as in the previous example. Reading these structures continues until this Driver finds null values in the first two date and time fields, indicating the end of the table, or until it reaches the limit of 255 operands.

Using resources from the previous example is designed to work only with blocks, so in case of using a single variable, create a block with only one variable.

Sequence of Steps Performed by this Driver

1. Checks bit 3 (three, table zeroed). If it is equal to 1 (one), exits or continues otherwise.
2. Writes 1 (one) to bit 0 (zero, asks permission).
3. Checks bit 1 (one). If it is equal to 1 (one), continues or checks once again otherwise. If it is still not authorized, exits. In this case, the PLC must handle the pending request, changing bit 0 (zero, asks permission) to 0 (zero).
4. Reads the first *n* + 2 variables, in which *n* is the block size.
5. In case the first 2 (two) values represent a valid date, these values are copied.
6. Repeats the previous step until reaching the end of the table or when null values are found in the first 2 (two) fields.
7. Clears the table, in case at least 1 (one) structure was read, by writing 1 (one) to bit 2 (two, clear).
8. This Driver returns those values to an **Elipse Software** application.

Data Blocks Acquired at Constant Intervals

Users must use this mode when the time interval among data on the table is fixed. Thus, users can save memory space in a PLC, only storing the initial moment of collecting. This mode can only be used when there is several data for a single variable. To do so, users must perform the next steps.

In a PLC, there must be a program that stores acquired data on a decimal table. This table's format must has the next structure.

```
Data 1 - HHMMSS
Data 2 - DDMMYY
Data 3 - time interval between acquisitions, in milliseconds
Data 4 - value 1
(...)
Data N - value N - 3
```

For example, on a table with 255 positions, users want to store several acquisitions of the same analog variable with constant intervals. Therefore, this table after completed contains the following configuration, assuming that the current date is February 18th, 1998, 09:00:00, with acquisitions at every second.

Example of data with constant timestamp intervals

TABLE POSITION	VALUE	TABLE POSITION	VALUE	TABLE POSITION	VALUE
0	090000	**7**	Data	**14**	Data
1	180298	**8**	Data	**15**	Data
2	1000	**9**	Data	**16**	Data
3	Data	**10**	Data	**17**	Data
4	Data	**11**	Data	**18**	Data
5	Data	**12**	Data	**19**	Data
6	Data	**13**	Data	**20**	Data

And so on, until the table is completely filled. To control access to this table, there are 4 (four) bits of an auxiliary memory in a PLC. This auxiliary operand is then provided during a Tag reading.

- **BIT 0 (ask_reading_permission)**: Performs a request to access the table to a PLC
- **BIT 1 (reading_allowed)**: Checks if the request was granted
- **BIT 2 (clear)**: This Driver writes 1 (one) to this bit and the PLC is then responsible for zeroing the table and turning this bit back to 0 (zero)
- **BIT 3 (table_zeroed)**: This bit indicates whether that table is zeroed

Reading data can only be performed by using a PLC Tag with the *N2* parameter equal to 13, where the *N3* parameter indicates the address of a decimal table, and the *N4* parameter indicates the address of an auxiliary operand that contains the control bits. Please check tables **Supported Data Types for PLC Tags** and **Supported Data Types for Block Tags** for more information. In case the data value read is different from 0 (zero), this Driver returns a list of event values to an **Elipse Software** application, each one with a timestamp equal to the table timestamp + (*N* - 3) × time interval, in which *N* is the position of the variable inside a table.

Sequence of Steps Performed by this Driver

1. Checks bit 3 (three, table zeroed). If it is set to 1 (one), exits or proceeds otherwise.
2. Writes 1 (one) to bit 0 (zero, ask permission).
3. Checks bit 1 (one) and, if it is set to 1 (one), continues. Otherwise, checks it once again. If the authorization is not yet granted, exits. In this case, the PLC must handle the pending request, changing bit 0 (zero, ask permission) to 0 (zero).
4. Reads the first 3 (three) variables with timestamp and interval.
5. In case the first 3 (three) values contain a valid date, stores them.

6. Reads the next table positions, in blocks of 12 units, storing the value together with the added timestamp of each interval, and so on. In case the first value of each reading is equal to 0 (zero), returns the last values read and zeroes the table, that is, only transfers values from a block with 12 variables if the first value is different from 0 (zero).

7. Zeroes the table, in case at least one structure was read, writing 1 (one) to bit 2 (two, clear).

8. This Driver returns the values to an **Elipse Software** application.

Unsolicited Messages (MNS)

Using unsolicited messages can significantly improve an application's performance, because there is no need to perform a polling on the variables of a PLC all the time, even when they are not changing their values. Instead, by using unsolicited messages, every time a variable changes its value, the PLC itself sends a message to this Driver with this new value which, by its turn, can update its database.

An application, by its turn, must configure its PLC and Block Tags similarly to the way regular Tags are configured, via polling, only changing its *N1* or *B1* parameter to 9999. This value indicates to this Driver that these Tag values must be read not from a PLC, but directly from this Driver's database, updated by unsolicited messages. Application Tags, therefore, keep working by polling, except that reading is performed directly from an image memory in this Driver, which does not generate communication with a PLC. As this operation is extremely fast, it is advisable to use a very short scan period, in milliseconds, which allows an immediate update of Tags in case of changing values by unsolicited messages. Reading unsolicited messages is only enabled in this Driver for operands of type **Memory**, **Memory Table**, **Auxiliary**, and **Octet**.

Example of a PLC Tag for Unsolicited Messages

- **N1**: 101 (Network equal to one and PLC equal to one)
- **N2**: 0 (zero, Memory)
- **N3**: 1 (one, M1 Memory)
- **N4**: 0 (zero, not used in this case)
- **Scan**: 500 (500 milliseconds)

To change this PLC Tag to read directly from this Driver's image memory, change the *N1* parameter to 9999. As this reading operation is much faster than polling, users can also change the scan period to a few milliseconds. The new configuration is the following:

- **N1**: 9999 (Reading of Image Memory)
- **N2**: 0 (zero, Memory)
- **N3**: 1 (one, M1 Memory)
- **N4**: 0 (zero, not used in this case)
- **Scan**: 10 (10 milliseconds)

Documentation of I/O Interfaces

This section contains the documentation of I/O Interfaces referring to the **AlnetIv2** Driver.

Configuration of a Driver

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

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

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

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

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

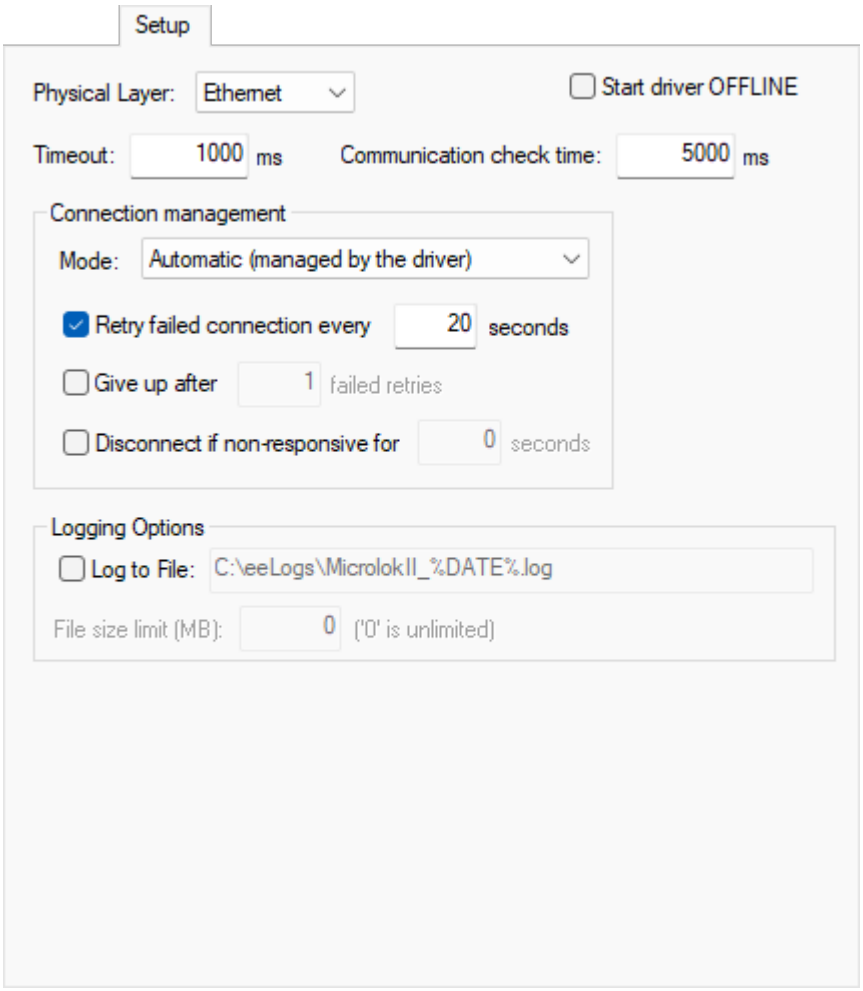
Configuration Dialog Box

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

Setup Tab

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

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



Setup tab

General options on the Setup tab

OPTION	DESCRIPTION
Physical Layer	Select the physical layer on a list. Available options are **Serial**, **Ethernet**, **Modem**, and **RAS**. The selected interface must be configured on its specific tab
Timeout	Configure a time-out, in milliseconds, for the physical layer. This is the amount of time an I/O interface waits to receive any byte from the reception's buffer
Communication check time	Set the time, in milliseconds, to define the interval at which communication is considered to be in an inactive state. As long as an I/O Driver receives valid data, its communication state is considered active. However, if during operation an I/O Driver does not receive valid data inside this period of time, the state is considered inactive. The communication state is shown in the **IO.CommunicationStatus** Tag
Start driver OFFLINE	Select this option so that a Driver starts in **Offline** mode or stopped. This means that the I/O interface is not created until this Driver is configured to **Online** mode by using a Tag in an application. This mode enables a dynamic configuration of an I/O interface at run time

Options on the Connection management group

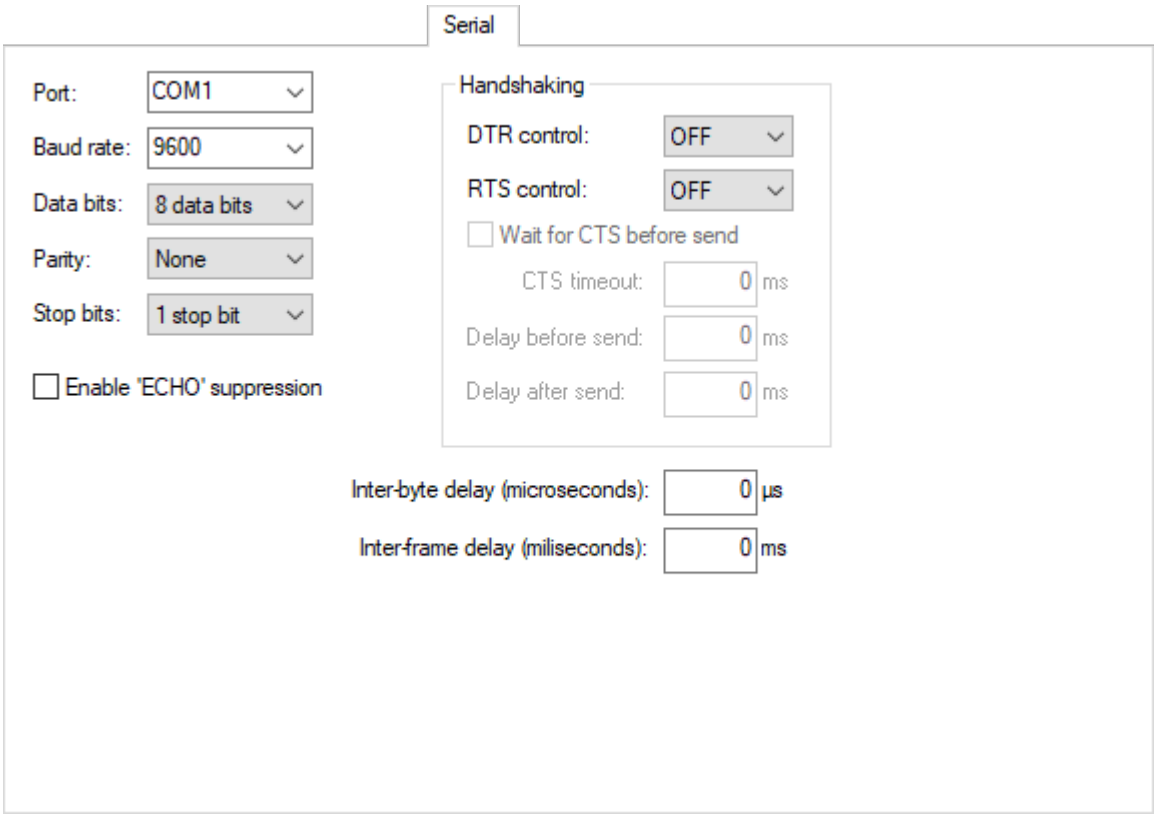
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection
Retry failed connection every ... seconds	Select this option to enable a Driver's connection retry in a certain interval, in seconds. If the **Give up after failed retries** option is not selected, this Driver keeps retrying until a connection is performed, or until the application is stopped
Give up after ... failed retries	Enable this option to define a maximum number of connection retries. When the specified number of consecutive connection retries is reached, a Driver goes to the **Offline** mode, assuming that a hardware problem was detected. If a Driver establishes a successful connection, the number of unsuccessful retries is cleared. If this new connection is lost, then the retry counter starts at zero
Disconnect if non-responsive for ... seconds	Enable this option to force a Driver to disconnect if no byte was received by the I/O interface during the specified time-out, in seconds. This time-out must be greater than the time-out configured in the **Timeout** option

Options on the Logging Options group

OPTION	DESCRIPTION
Log to File	Enable this option and configure the name of a file to write a log. Log files can be large, so use this option for short periods of time, only for testing and debugging purposes. If the **%PROCESS%** macro is used in the log file name, it is replaced by the identifier of the current process. This option is particularly useful when using several instances of the same Driver in **Elipse E3**, thus allowing each instance to generate a separate log file. For example, when configuring this option with value "c:\e3logs\drivers\sim_%PROCESS%.log", it generates a file named c:\e3logs\drivers\sim_00000FDA.log for process **0FDAh**. Users can also use the **%DATE%** macro in the file name. In this case a log file is generated every day, in the format **aaaa_mm_dd**. For example, when configuring this option with value "c:\e3logs\drivers\sim_%DATE%.log", it generates a file named c:\e3logs\drivers\sim_2005_12_31.log in 12/31/2005 and a file named c:\e3logs\drivers\sim_2006_01_01.log in 01/01/2006. Similarly, the **%DATE_HOUR%** macro generates one log file per hour, in the format **aaaa_mm_dd_hh**
File size limit (MB)	Configure the log file size limit, in megabytes. A value equal to 0 (zero) means that there is no size limit for the log file

Serial Tab

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



Serial tab

General options on the Serial tab

OPTION	DESCRIPTION
Port	Select a serial port on the list, from **COM1** to **COM4**, or type the name of a serial port in the format **COM*n***, such as "COM15". When typing the name of a serial port manually, the dialog box only accepts names of serial ports starting with the expression "COM"
Baud rate	Select a baud rate on the list (**1200**, **2400**, **4800**, **9600**, **19200**, **38400**, **57600**, or **115200**) or type a baud rate, such as 600
Data bits	Select 7 (seven) or 8 (eight) data bits on the list
Parity	Select a parity on the list. The available options are **None**, **Even**, **Odd**, **Mark**, or **List**
Stop bits	Select the number of stop bits on the list. The available options are **1**, **1.5**, or **2** stop bits
Enable 'ECHO' supression	Enable this option to remove the echo received after the I/O Interface sends data via serial port. If this echo is not equal to the bytes just sent, then the I/O Interface aborts communication
Inter-byte delay (microseconds)	Defines a delay between each byte transmitted by the I/O Interface, in millionths of a second, that is, 1000000 is equal to a second. This option must be used with small delays of less than a millisecond
Inter-frame delay (milliseconds)	Defines a delay between packets sent or received by the I/O Interface, in thousandths of a second, that is, 1000 is equal to a second. This delay is applied if the I/O Interface

OPTION	DESCRIPTION
	sends two consecutive packets, or between a received packet and the next sending

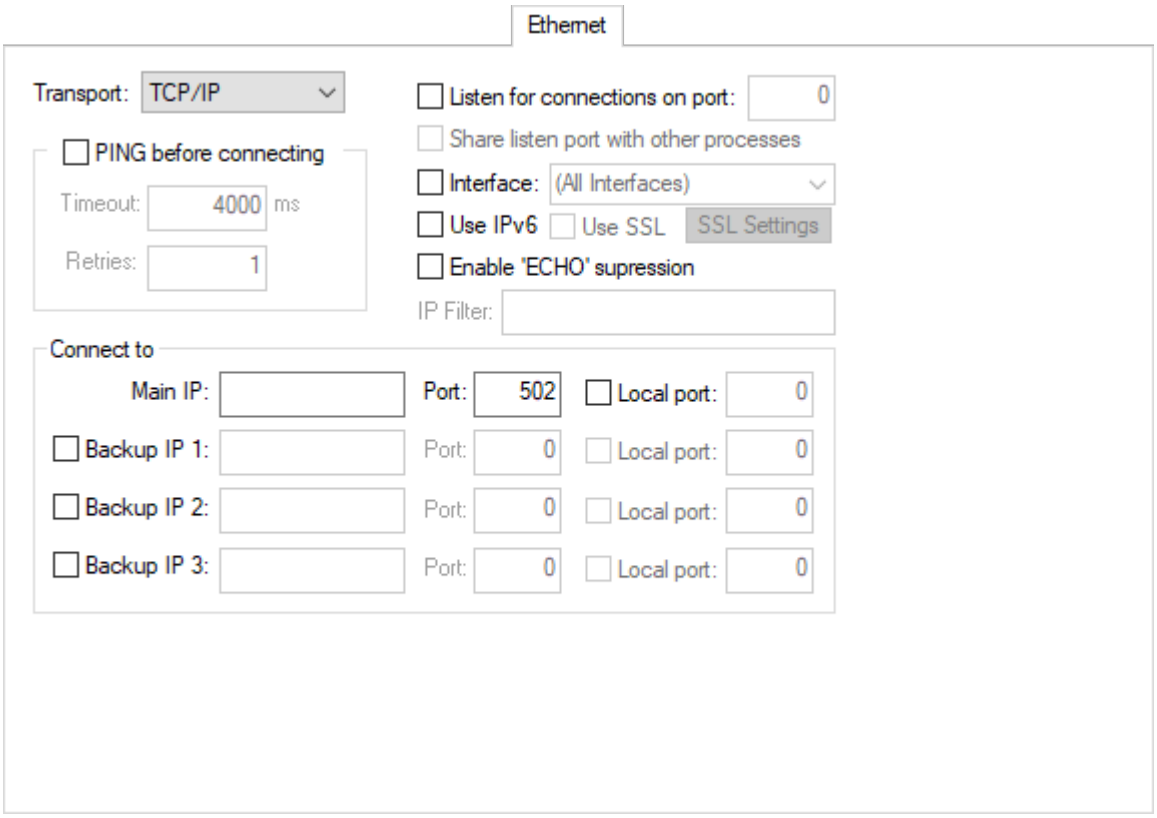
The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process, that is, it controls when data can be sent or received via serial line. Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works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

Available options on the Handshaking group

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

Ethernet Tab

Use this tab to configure parameters of an **Ethernet** Interface. These parameters, except port configurations, must also be configured for use in the **RAS** Interface.



Ethernet tab

Available options on the Ethernet tab

OPTION	DESCRIPTION
Transport	Select the value **TCP/IP** for a TCP socket (*stream*) or select the value **UDP/IP** to use a UDP socket (*connectionless datagram*)
Listen for connections on port	Use this option to wait for new connections in a specific IP port, common in Slave Drivers. If this option remains unselected, a Driver connects to the address and port specified in the **Connect to** option
Share listen port with other processes	Select this option to share the listening port with other Drivers and processes
Interface	Select the local network interface, identified by its IP address, that a Driver uses to establish and receive connections, or select the value **(All Interfaces)** to allow connection in any network interface
Use IPv6	Select this option to force a Driver to use addresses in **IPv6** format on all Ethernet connections. Leave this option deselected to use the **IPv4** format
Enable 'ECHO' suppression	Enable this option to remove the echo from received data. An echo is a copy of sent data, which can be returned before a reply message
IP Filter	List of restricted or allowed IP addresses from where a Driver accepts connections (*Firewall*). Please check the **IO.Ethernet.IPFilter** property for more information
PING before connecting	Enable this option to execute a **ping** command, that is, to check whether a device can be reached on a network, for a device before trying a socket connection. This is a quick way

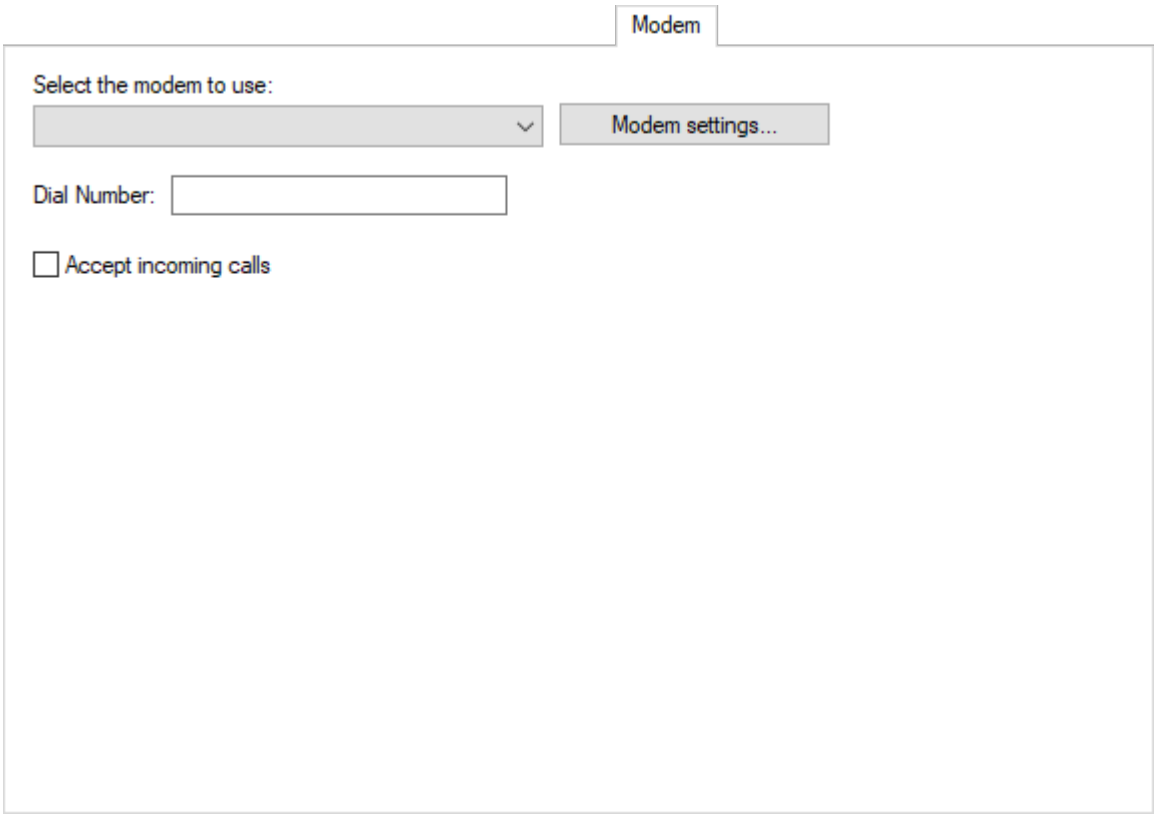
OPTION	DESCRIPTION
	of determining a successful connection before trying to open a socket with a device. The time-out of a connection with a socket can be very high. The available options are: • **Timeout**: Specify the number of milliseconds to wait for a reply from a **ping** command. Users must use a **ping** command to check the normal reply time, configuring this option for a value above that average. Usually this value can be configured between 1000 and 4000 milliseconds, that is, between 1 (one) and 4 (four) seconds • **Retries**: Number of retries of a **ping** command, not counting the first attempt. If all attempts fail, then the socket connection is aborted

Available options on the Connect to group

OPTION	DESCRIPTION
Main IP	Type the IP address of a remote device. Users can use an IP address separated by dots, as well as a URL. In case of a URL, a Driver uses the available DNS service to map that URL to an IP address, such as "192.168.0.13" or "Server1"
Port	Type the IP port of a remote device, between 0 (zero) and 65535
Local port	Select this option to use a fixed local IP port when connecting to a remote device
Backup IP 1, 2, and 3	Indicate the IP address, the IP port, and the fixed local IP port of up to 3 (three) backup addresses of a remote device

Modem Tab

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



Modem tab

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

Available options on the Modem tab

OPTION	DESCRIPTION
Select the modem to use	Select a modem on the list of available modems on the computer. If the value **Default modem** is selected, then the first available modem is used. Selecting this option is recommended specially when an application is used on another computer
Modem settings	Click to open the configuration window of the selected modem
Dial Number	Type a default number for dialing. This value can be changed at run time. Users can use the **w** character to represent a pause or a waiting time for a dial tone. For example, "0w33313456" dials the number 0 (zero), waits, and then dials the number "33313456"
Accept incoming calls	Enable this option so that a Driver answers the phone when receiving an external call. To use this option, users must configure the **Connection management** option on the **Setup** tab to the value **Manual**

RAS Tab

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

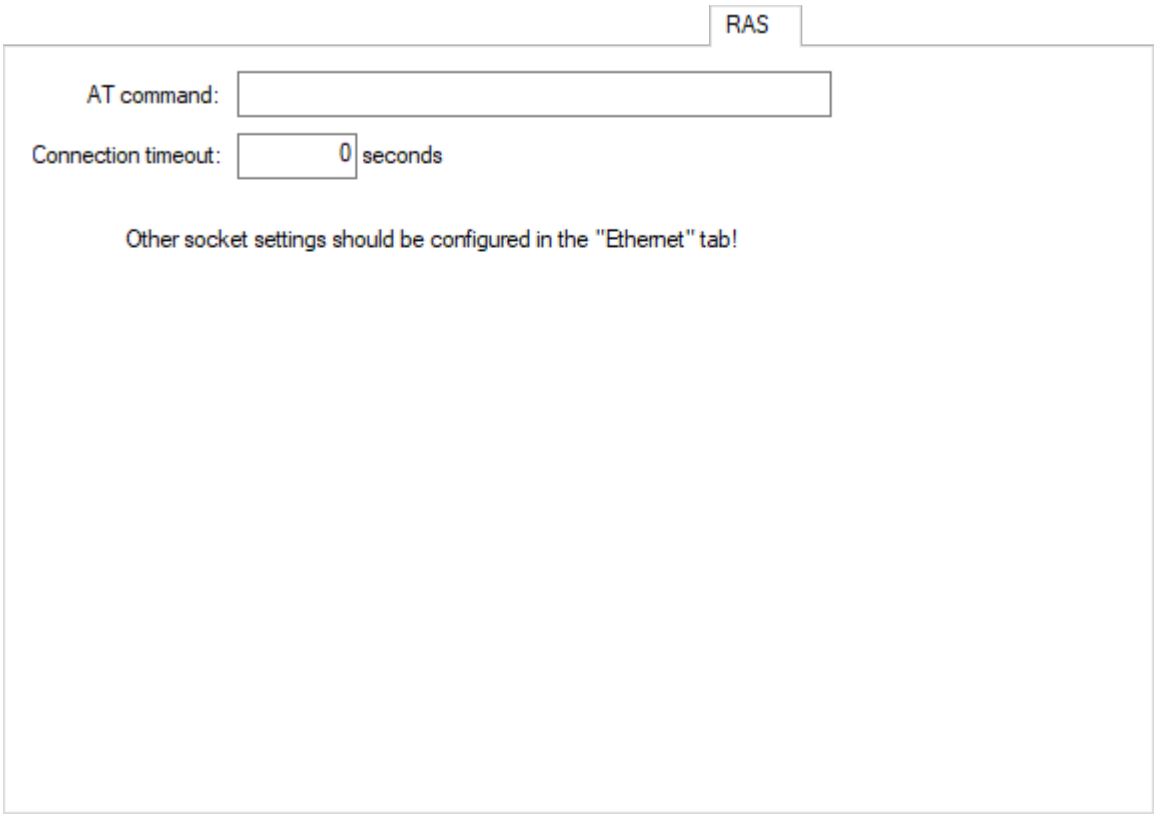
A **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

When connecting to a RAS device, first the I/O Interface**IOKit** connects to the socket on the IP address and port configured on the **Ethernet** tab. After opening the socket, the following initialization or connection steps are performed:

1. Clears the socket, that is, removes any **TELNET** greeting message received from a RAS device.
2. Sends an **AT** dial message, in **ASCII** format, in the socket.
3. Waits for a **CONNECT** reply.
4. If the time-out expires, the connection is aborted.
5. If the **CONNECT** reply is received within the time-out, the socket is available for communication with a device, that is, the connection was established.

If step 5 (five) is successful, then the socket behaves as a normal socket, with the RAS device working as a router between a Driver and the device. Bytes sent by a Driver are received by the RAS device and sent to the destination device using a modem. Bytes received by the modem's RAS device are sent back to a Driver using the same socket.

After establishing a connection, the **RAS** interface monitors data received by a Driver. If a "NO CARRIER" **String** is found, the socket is closed. If the RAS device does not send a **NO CARRIER** signal, the **RAS** Interface cannot detect when the modem connection between the RAS device and the final I/O device fails. To recover from this failure, users are strongly advised to enable the **Disconnect if non-responsive** option on the **Setup** tab.



RAS tab

Available options on RAS tab

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

General Configurations

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

I/O Tags

General I/O Interfaces Tags (N2/B2 = 0)

The Tags described next are provided for all supported I/O Interfaces.

IO.CommunicationStatus

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

This Tag informs the communication status of a Driver. It indicates how communication works relative to receiving valid data within a time period arbitrated in the configuration. For more information, please check topic **Setup Tab**. Possible values are **0 - Inactive communication**: The Driver did not receive valid data or stopped receiving data after *n* milliseconds, as configured in the properties window, or **1 - Active communication:** The Driver is receiving valid data.

IO.IOKitEvent

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

This Block returns Driver events generated by several sources in I/O Interfaces. The **TimeStamp** property of this Block represents the moment this event occurred. The Block Elements are the following:

- **Element 0**: Type of event. Possible values are **0**: Information, **1**: Warning, or **2**: Error
- **Element 1**: Source of an event. Possible values are **0**: Driver (specific of a Driver), **-1**: IOKit (generic events of I/O Interfaces), **-2**: **Serial** Interface, **-3**: **Modem** Interface, **-4**: **Ethernet** Interface, or **-5**: **RAS** Interface
- **Element 2**: Error number, specific for each source of event

- **Element 3**: Message of an event, a **String** specific for each event

NOTE

A Driver keeps a maximum number of 100 events internally. If additional events are reported, older events are discarded.

IO.PhysicalLayerStatus

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	2 (two)
String Configuration	IO.PhysicalLayerStatus

This Tag indicates the status of a physical layer. Possible values are the following:

- **0**: Physical layer stopped, that is, a Driver is in **Offline** mode, the physical layer failed when initializing, or exceeded the maximum number of reconnection attempts
- **1**: Physical layer started but not connected, that is, a Driver is in **Online** mode but the physical layer is not connected. If the **Connection management** option is configured with the value **Automatic**, the physical layer can be connecting, disconnecting, or waiting for a reconnection attempt. If the **Connection management** option is configured with the vlaue **Manual**, then the physical layer remains in this status until forced to connect
- **2**: Physical layer connected, that is, the physical layer is ready for use. This **DOES NOT** mean a device is connected, only that the access layer is working

IO.SetConfigurationParameters

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

Use this Tag to change any property of a Driver's configuration dialog box at run time.

This Tag works only while a Driver is in **Offline** mode. To start a Driver in **Offline** mode, select the **Start driver OFFLINE** option on that Driver's configuration dialog box. Users can write to a PLC Tag or to a Block Tag containing the parameters to change. Writing individual Block Elements is not supported, the whole Block must be written at once.

In **Elipse SCADA**, users must use a Block Tag. Every parameter to configure uses two Block Elements. For example, if users want to configure 3 (three) parameters, then the size of the Block must be 6 (six, 3 × 2). The first Element is the property's name, as a **String**, and the second Element is the property's value, according to the next example.

```
// 'Block' must be a Block Tag with automatic reading,
// scan reading, and automatic writings disabled.
// Configure all parameters
Block.element001 = "IO.Type" // Parameter 1
Block.element002 = "Serial"
Block.element003 = "IO.Serial.Port" // Parameter 2
Block.element004 = 1
Block.element005 = "IO.Serial.BaudRate" // Parameter 3
Block.element006 = 19200
// Writes the whole Block
Block.Write()
```

When using **Elipse E3**, the ability to create arrays at run time allows using an I/O Tag as well as a Block Tag. Users can use the **Write** method of a Driver to send the parameters directly to that Driver, without creating a Tag, according to the next example.

```
Dim arr(6)
' Configure all array elements
arr(1) = "IO.Type"
arr(2) = "Serial"
arr(3) = "IO.Serial.Port"
arr(4) = 1
arr(5) = "IO.Serial.BaudRate"
arr(6) = 19200
' There are two methods to send parameters
' Method 1: Using an I/O Tag
tag.WriteEx arr
' Method 2: Without using a Tag
Driver.Write -1, 0, 0, 3, arr
```

A variation of the previous example uses a bidimensional array.

```
Dim arr(10)
' Configure all array elements. Notice the array was resized
' to 10 elements. Empty array elements are ignored by a Driver
arr(1) = Array("IO.Type", "Serial")
arr(2) = Array("IO.Serial.Port", 1)
arr(3) = Array("IO.Serial.BaudRate", 19200)
Driver.Write -1, 0, 0, 3, arr
```

A Driver does not validate parameter names or passed values, therefore be careful when writing parameters and values. The **Write** method fails if the configuration array is incorrectly created. Users can check the log of a Driver or use the *writeStatus* parameter of the **WriteEx** method to find out the exact cause of an error.

```
Dim arr(10), strError
arr(1) = Array("IO.Type", "Serial")
arr(2) = Array("IO.Serial.Port", 1)
arr(3) = Array("IO.Serial.BaudRate", 19200)
If Not Driver.WriteEx -1, 0, 0, 3, arr, , , strError Then
  MsgBox "Failed configuring Driver parameters: " + strError
End If
```

IO.WorkOnline

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

This Tag informs the current status of a Driver and allows starting or stopping the physical layer. Possible values are the following:

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

In the next example, using **Elipse E3**, a Driver is configured to **Offline** mode, its COM port is changed, and then configured to **Online** mode again.

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

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

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

IMPORTANT

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

Properties

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

IO.ConnectionMode

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

IO.GiveUpEnable

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

IO.GiveUpTries

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

IO.InactivityEnable

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

IO.InactivityPeriodSec

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

IO.RecoverEnable

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

IO.RecoverPeriodSec

Delay time between two connection attempts, in seconds.

NOTE

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

IO.StartOffline

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

NOTE

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

IO.TimeoutMs

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

IO.Type

Defines the type of physical interface used by a Driver. Possible values are the following:

- **N or None**: Does not use a physical interface, that is, a Driver must provide a customized interface
- **S or Serial**: Uses a local serial port (COM*n*)
- **M or Modem**: Uses a local modem, internal or external, accessed via TAPI (*Telephony Application Programming Interface*)
- **E or Ethernet**: Uses a TCP/IP or UDP/IP socket
- **R or RAS**: Uses a **RAS** (*Remote Access Server*) Interface. A Driver connects to a RAS device using the **Ethernet** Interface and then sends an **AT** (*dial*) command

Statistical Configuration

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

I/O Tags

Tags of I/O Interface Statistics (N2/B2 = 0)

The Tags described next display statistics for all I/O Interfaces.

IO.Stats.Partial.BytesRecv

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1101
Configuration by String	IO.Stats.Partial.BytesRecv

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

IO.Stats.Partial.BytesSent

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1100
Configuration by String	IO.Stats.Partial.BytesSent

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

IO.Stats.Partial.TimeConnectedSeconds

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1102
Configuration by String	IO.Stats.Partial.TimeConnectedSeconds

This Tag returns the number of seconds a Driver is connected in the current connection or 0 (zero) if a Driver is disconnected.

IO.Stats.Partial.TimeDisconnectedSeconds

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1103
Configuration by String	IO.Stats.Partial.TimeDisconnectedSeconds

This Tag returns the number of seconds a Driver is disconnected since the last connection ended or 0 (zero) if a Driver is connected.

IO.Stats.Total.BytesRecv

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1001
Configuration by String	IO.Stats.Total.BytesRecv

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

IO.Stats.Total.BytesSent

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1000
Configuration by String	IO.Stats.Total.BytesSent

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

IO.Stats.Total.ConnectionCount

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1004
Configuration by String	IO.Stats.Total.ConnectionCount

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

IO.Stats.Total.TimeConnectedSeconds

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1002
Configuration by String	IO.Stats.Total.TimeConnectedSeconds

This Tag returns the number of seconds a Driver remained connected since it was loaded.

IO.Stats.Total.TimeDisconnectedSeconds

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1003
Configuration by String	IO.Stats.Total.TimeDisconnectedSeconds

This Tag returns the number of seconds a Driver remained disconnected since it was loaded.

Properties

Currently, there are no properties defined specifically to display I/O Interface statistics at run time.

Ethernet Interface Configuration

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

I/O Tags

Tags of an Ethernet Interface (N2/B2 = 4)

The Tags described next allow controlling and identifying an **Ethernet** Interface at run time and they are also valid when the **RAS** Interface is selected.

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

IO.Ethernet.IPSelect

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

Indicates the active IP address. Possible values are **0**: The main IP address is selected, **1**: The first alternative or backup IP address is selected, **2**: The second alternative or backup IP address is selected, or **3**: The third alternative or backup IP address is selected.

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

During the connection process, if the active IP address is not available, the I/O Interface tries to connect using the other IP address. If the connection with the alternative IP address works, it is configured as the active IP address (automatic switchover).

To force a manual switchover, write values from 0 (zero) to 3 (three) to this Tag. This forces a reconnection with the specified IP address (**0**: Main address or **1, 2, 3**: Alternative address) if a Driver is currently connected. If a Driver is disconnected, this Tag configures the active IP address for the next attempt to connect.

IO.Ethernet.IPSwitch

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

Any value written to this Tag forces a manual switchover. If the main IP address is active, then the first alternative or backup IP address is activated, and so on for all alternative IP addresses and returning to the main address until a connection is established.

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

IO.Ethernet.SocketState

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

The Value property of this Tag corresponds to socket states as a map of bits:

- **Bit 0**: 0 (zero, not listening) or 1 (one, listening)
- **Bit 1**: 0 (zero, disconnected) or 1 (one, connected)

Properties

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

NOTE

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

IO.Ethernet.AcceptConnection

Configure to False if a Driver must not accept external connections, that is, if a Driver behaves as a master, or configure to True to enable the reception of connections, that is, if a Driver behaves as a slave.

IO.Ethernet.BackupEnable[2,3]

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

IO.Ethernet.BackupIP[2,3]

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

IO.Ethernet.BackupLocalPort[2,3]

Local port number to be used when connecting to an alternative IP address of a remote device. Used only if **IO.Ethernet.BackupLocalPortEnable** is equal to True.

IO.Ethernet.BackupLocalPortEnable[2,3]

Configure to True to force the use of a specific local port when connecting to an alternative or backup IP address or configure to False to use any available local port.

IO.Ethernet.BackupPort[2,3]

Port number of an alternative or backup IP address of a remote device, used with the **IO.Ethernet.BackupIP** property.

IO.Ethernet.IPFilter

List with a comma-separated IPv4 or IPv6 addresses, which defines from which addresses a Driver accepts or blocks connections. Users can use asterisks, such as "192.168.*.*", or intervals, such as "192.168.0.41-50", in any part of IP addresses. To block an IP address or a range of IP addresses, use the tilde ("~") character at the beginning of the address, according to the next examples:

- **192.168.0.24**: Accepts only connections from IPv4 address 192.168.0.24
- **192.168.0.41-50**: Accepts connections from IPv4 addresses in the interval between 192.168.0.41 and 192.168.0.50
- **192.168.0.***: Accepts connections from IPv4 addresses in the interval between 192.168.0.0 and 192.168.0.255
- **fe80:3bf:877::*:* (expands to fe80:03bf:0877:0000:0000:0000:*:*)**: Accepts connections from IPv6 addresses in the interval between fe80:03bf:0877:0000:0000:0000:0000:0000 and fe80:03bf:0877:0000:0000:0000:ffff:ffff
- **192.168.0.10, 192.168.0.15, 192.168.0.20**: Accepts connections from IPv4 addresses 192.168.0.10, 192.168.0.15, and 192.168.0.20
- **~192.168.0.95, 192.168.0.***: Accepts connections from IPv4 addresses in the interval between 192.168.0.0 and 192.168.0.255, except the IPv4 address 192.168.0.95

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

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

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

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

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

IO.Ethernet.ListenIP

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

IO.Ethernet.ListenPort

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

IO.Ethernet.MainIP

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

IO.Ethernet.MainLocalPort

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

IO.Ethernet.MainLocalPortEnable

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

IO.Ethernet.MainPort

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

IO.Ethernet.PingEnable

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

IO.Ethernet.PingTimeoutMs

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

IO.Ethernet.PingTries

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

IO.Ethernet.ShareListenPort

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

IO.Ethernet.SupressEcho

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

IO.Ethernet.Transport

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

IO.Ethernet.UseIPv6

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

Modem Interface Configuration

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

I/O Tags

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

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

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

IO.TAPI.ConnectionBaudRate

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	5 (five)
String Configuration	IO.TAPI.ConnectionBaudRate

Indicates a baud rate value for the current connection. If a modem is not connected, returns the value 0 (zero).

IO.TAPI.Dial

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

Write any value to this Tag to force a **Modem** Interface to start a call. This is an asynchronous command, only starting the call process. Users can monitor the **IO.TAPI.IsModemConnected** Tag to detect when a call is established.

IO.TAPI.HangUp

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

Any value written to this Tag hangs the current call up.

NOTE

Use this command only when managing the physical layer manually or when explicitly trying to force a Driver to restart the communication. If the physical layer is configured for automatic reconnection, a Driver immediately tries to reestablish the connection.

IO.TAPI.IsModemConnected

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

This Tag indicates the status of a modem connection. Possible values are **0**: The modem is not connected, but it may be performing or receiving an external call or **1**: The modem is connected and a Driver completed or received an external call successfully. While it is in this status, the physical layer can send or receive data.

IO.TAPI.IsModemConnecting

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	6 (six)
String Configuration	IO.TAPI.IsModemConnecting

This Tag indicates the status of a modem connection, with more details than the **IO.TAPI.IsModemConnected** Tag. Possible values are **0**: Modem is not connected, **1**: Modem is connecting, that is, performing or receiving an external call, **2**: Modem is connected. While in this status, the physical layer can send or receive data, or **3**: Modem is disconnecting the current call.

IO.TAPI.ModemStatus

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	2 (two)
String Configuration	IO.TAPI.ModemStatus

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

- **"No status!"**: The **Modem** Interface was not open yet or was already closed
- **"Modem initialized OK!"**: The **Modem** Interface was initialized successfully
- **"Modem error at initialization!"**: A Driver could not initialize modem's line. Check that Driver's log file for more details
- **"Modem error at dial!"**: A Driver could not start or accept a call
- **"Connecting..."**: A Driver started a call successfully, and is currently processing that call
- **"Ringing..."**: Indicates that the modem is receiving an external call, but it did not accepted it yet
- **"Connected!"**: A Driver connected successfully, that is, completed or accepted an external call
- **"Disconnecting..."**: A Driver is turning the current call off
- **"Disconnected OK!"**: A Driver turned the current call off
- **"Error: no dial tone!"**: A Driver aborted a call because the available line signal was not detected
- **"Error: busy!"**: A Driver aborted a call because the line was busy
- **"Error: no answer!"**: A Driver aborted a call because no answer was received from the other modem
- **"Error: unknown!"**: Current call was aborted because of an unknown error

IO.TAPI.PhoneNumber

Type of Tag	I/O Tag
Type of Access	Reading or Writing
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	0 (zero)
String Configuration	IO.TAPI.PhoneNumber

This Tag is a **String** that reads or changes the telephone number used by the **IO.TAPI.Dial** Tag. When changing this Tag, the new value is used only on the next **Dial** command.

Properties

These properties control the configuration of a **Modem** (TAPI) Interface.

IO.TAPI.AcceptIncoming

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

IO.TAPI.ModemID

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

NOTE

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

IO.TAPI.PhoneNumber

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

RAS Interface Configuration

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

I/O Tags

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

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

Properties

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

NOTE

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

IO.RAS.ATCommand

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

IO.RAS.CommandTimeoutSec

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

Serial Interface Configuration

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

I/O Tags

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

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

Properties

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

IO.Serial.Baudrate

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

IO.Serial.CTSTimeoutMs

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

IO.Serial.DataBits

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

IO.Serial.DelayAfterMs

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

IO.Serial.DelayBeforeMs

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

IO.Serial.DTR

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

IO.Serial.InterbyteDelayUs

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

IO.Serial.InterframeDelayMs

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

IO.Serial.Parity

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

IO.Serial.Port

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

IO.Serial.RTS

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

IO.Serial.StopBits

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

IO.Serial.SupressEcho

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

IO.Serial.WaitCTS

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

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
2.4.10	09/01/2025	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 37935*).
2.4.9	10/02/2020	C. Mello	• Added support for **32-Bit Integer** operands (*Case 29471*). • Changed the *B2* parameter of the Data Block with Timestamp Tag from 6000 to 5005.
2.4.8	09/27/2019	C. Mello	• Driver ported to Visual Studio 2017 (*Case 27437*).
2.4.7	09/27/2013	A. Quites	• Adjustments in the validation of Tag parameters (*Case 14706*).

VERSION	DATE	AUTHOR	COMMENTS
			• Fixed an error when reading **IOKit** library's Internal Tags (*Case 13585*). • Driver ported to **IOKit** library version **2.0** (*Case 13654*).
2.3.1	10/07/2008	A. Quites	• Added an unsigned integer data type (*Case 9555*).
2.2.1	10/31/2007	A. Quites	• Fixed an error when writing bits of **Auxiliary** and **Octet** operands with the *N4* parameter greater than 0 (zero) (*Case 8887*).
2.1.1	03/21/2007	A. Quites	• Fixed an error when reading data blocks with timestamps, acquired by the simultaneous acquisition mode, in which this Driver was not sending a command to clear the table after reading, causing this data to be read more than once (*Case 7374*).
2.0.1	07/28/2006	A. Quites	• Original version for **IOKit** library, unifying **AL2000**, **AL2000 Timestamp**, **AL2000MNS**, and **AL2000-Modem** Drivers.
1.0.1		R. Haetinger	• All releases of Drivers for Altus AL2000 published before revision control.

Headquarters
Rua Mostardeiro, 322/Cj. 902, 1001 e 1002
90510-002 — Porto Alegre — RS
Phone: (+55 51) 3346-4699
Fax: (+55 51) 3222-6226
E-mail: elipse-rs@elipse.com.br

Branch in Taiwan
9F., No.12, Beiping 2nd St., Sanmin Dist.
807 — Kaohsiung City — Taiwan
Phone: (+886 7) 323-8468
Fax: (+886 7) 323-9656
E-mail: evan@elipse.com.br

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

www.elipse.com.br

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

Gartner 2014 Cool Vendor

OPC FOUNDATION MEMBER

Gartner, Cool Vendors in Brazil 2014, April 2014.
Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability of fitness for a particular purpose.

Microsoft Partner
Gold Independent Software Vendor (ISV)