Atos APR03 Driver

File Name	APR03.dll
Manufacturer	Atos Automação Industrial Ltda.
Devices	MPC1200, MPC1600, MPC2000, MPC2002, MPC4004, EXPERT, 4004.72, and 4004.72R modules
Protocol	APR03
Version	3.0.2
Latest Update	09/01/2025
Platform	Win32
Dependencies	IOKit version 2.0 or later
Superblock Readings	Yes
Level	0

Introduction

This is Atos APR03 Driver for communication between **Elipse Software** systems and Atos controllers implementing APR03 protocol.

Preparing a Device

This section contains information about **[P]** parameters and extra settings of this Driver.

[P] Parameters for Driver Configuration

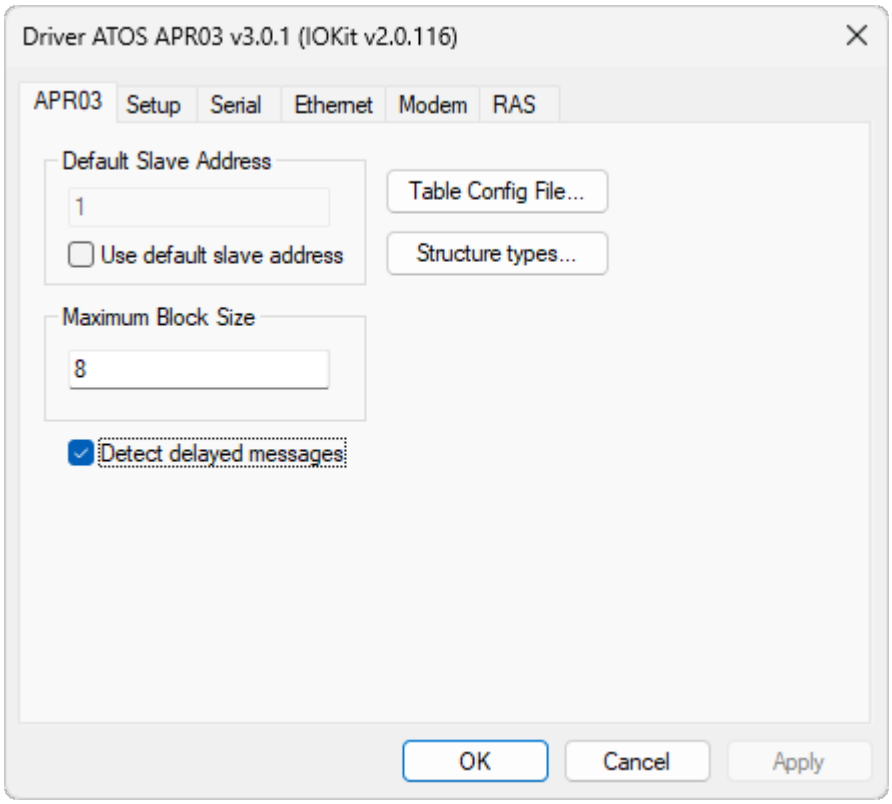
P1	Not used
P2	Not used
P3	Not used
P4	Not used

This Driver does not use **[P]** configuration parameters. All settings of this Driver must be performed on the configuration window.

The **APR03** tab on the configuration window displays specific settings for this Driver. The other tabs allow configuring communication parameters of the **IOKit** library. For more information about the configuration of the **IOKit** library, please check topic **Documentation of I/O Interfaces**.

APR03 Tab

To configure this Driver, use the **APR03** tab, shown on the next figure.



APR03 tab

Available options on the APR03 tab

OPTION	DESCRIPTION
Default Slave Address	Indicates the value of the default slave address. This option is enabled only if the **Use default slave address** option is selected
Use default slave address	Enables using the value defined in the **Default Slave Address** option as the destination address of a command for all Tags, regardless of the value configured in the *N1* or *B1* parameters
Maximum Block Size	This option is only used in **Elipse E3**, **Elipse Power**, or **Elipse Water** if a Tag's **EnableReadGrouping** property (Superblocks) is enabled. It informs the maximum number of data bytes in each reading block. Adequate values for each device model are described on the table **Limits for each I/O Block**
Detect delayed messages	This option allows this Driver to not return an error when detecting a delayed message. Please check the next **note** for more information about this option
Table Config File	Opens a **configuration file with tables for reading data by events**
Structure types	Opens a **configuration file for Struct data types**

Limits for each I/O Block

MODEL	LIMIT
Devices from series MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, EXPERT, and all controllers from series MPC2200	8 (eight) bytes
Devices MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, MPC4004.09T, and module 4004.72	16 bytes
Module 4004.72R	32 bytes

NOTE

The **Detect delayed messages** option is useful when the TCP/IP protocol (Ethernet) is used as the physical layer of **IOKit** library. In this case, it is a common situation when unwanted delays occur in a device's response, specially if the physical connection includes radio links. Therefore, this Driver may send a first request and does not receive a response in the expected time, thus generating a time-out error. Instead of sending a new request, this Driver can receive the delayed message from the previous request. If this option is not enabled, a syncing can happen, thus all subsequent requests get the delayed return from the previous request as a response. When this option is enabled, this Driver does not return an error immediately after detecting a delayed message, but executes a new attempt to get the correct response. If a new message is received before a time-out occurs, and that response is valid, it is returned to the application. Otherwise, this Driver starts discarding all bytes received up until the time-out, thus clearing the I/O channel and preventing the propagation of errors.

Configuration File for Reading Data by Event

To edit a configuration file for reading data by event, click **Table Config File** on the **APR03** tab. In this file, users can define the addresses of variables and pointers to control the reading of data tables, as well as the format of each table. The fields that must be defined in this file are described on the next table.

Fields of a configuration file

FIELD	DESCRIPTION
RECORD POINTER ADDRESS	Value of the recording pointer's address, in decimal
TABLE STATUS ADDRESS	Value of the table's status address, in decimal
ACQUISITION STATUS ADDRESS	Value of the acquisition's status address, in decimal
TABLE	Number of a table format to define, between 0 (zero) and 65535. There is no need to use a colon between the keyword and the number
REGISTERS	Maximum number of records or structures a table must support in its buffer. This value define the size of a circular buffer reserved for a table
START ADDRESS	Initial address of a table, in decimal, in which the circular buffer reserved for a table starts
PACKET SIZE	Maximum size of a data block that can be transferred at each communication with a device, in bytes. This value

FIELD	DESCRIPTION
	depends on the device used. The larger, the less communication is needed to read all data
STRUCT	Defines a structure for each record of a table. Its members must be defined by integer values equivalent to **data types interpreted by this Driver**

Users can add comment lines to a file by using the // characters, according to the pattern used by the C language. The topic **Reading Data by Event** analyzes this feature in details.

Configuration File for Struct Data Types

To edit **Struct** data types, that is, the *N2* or *B2* parameters equal to 20, click **Structure types** on the **APR03** tab. The declaration of these structures must follow the next syntax.

```
STRUCT TYPE
{
  Element1,
  Element2,
  Element3,
  [...]
}
```

The fields that must be defined in this file are described on the next table.

Fields of a configuration file

FIELD	DESCRIPTION
TYPE	Defines the number of a **Struct** data type. In I/O Tags, this value is used in the *N4* or *B4* parameters
Element	Elements must be separated by commas and between braces, after defining a **TYPE** field, and define the structure of each record on a table. Members must be defined by integer values equivalent to **data types interpreted by this Driver**. **String** data types, that is, with the *N2* parameter equal to 19, are not accepted as members of structures

Reading **Struct** data types must be performed by a Block Tag in which each Block Element corresponds to a member of that structure.

In addition to declaring structures, this file can also contain comments, preceded by the // characters, according to the pattern used by the C language. The next code contains an example of a structure.

```
// Struct accessed by Tags with the N/B parameters
// equal to 20 and N4/B4 equal to 1 (one, type)
STRUCT 1
{
  1, // WORDBCDLH
  0, // WORDBCDHL
  3, // WORDHEXAHL
}
```

Tag Reference

This section contains information about **[N/B] addressing parameters for PLC and Block Tags**, as well as about **reading data by event**.

[N/B] Addressing Parameters for PLC and Block Tags

N1/B1	Address of a PLC
N2/B2	Data type of a variable
N3/B3	Absolute address, in decimal
N4/B4	Not used

By using Tags of this Driver, users can read from and write to almost all types of variables from Atos devices.

Data Types Interpreted by this Driver

Data types

N2/B2	DATA TYPE	RANGE OF VALUES	BYTES
0	**BCD Word** (Hi - Lo)	Between 0 (zero) and 9999	2 (two)
1	**BCD Word** (Lo - Hi)	Between 0 (zero) and 9999	2 (two)
2	**Hex Word** (Lo - Hi)	Between 0 (zero) and 65535	2 (two)
3	**Hex Word** (Hi - Lo)	Between 0 (zero) and 65535	2 (two)
4	**Byte**	Between 0 (zero) and 255	1 (one)
5	**Modified Word** (Hi × 100 + Lo)	Between 0 (zero) and 25755	2 (two)
6	**Digital Byte**, that is, PLC byte XXXXXXX0 equal to 1 (one) or PLC byte XXXXXXX1 equal to 0 (zero)	0 (zero) or 1 (one)	1 (one)
7	**Inverted Word**, that is, each bit 0 (zero) is equal to 1 (one) and each bit 1 (one) is equal to 0 (zero)	Between 0 (zero) and 65535	2 (two)
8	**Digital Byte**, writing the last bit, that is, 1 (one) is equal to PLC byte XXXXXXX0 or 0 (zero) is equal to PLC byte XXXXXXX1	0 (zero) or 1 (one)	1 (one)
9	**BCD DWord**, WHi (Hi - Lo) - WLo (Hi - Lo)	Between 0 (zero) and 99999999	4 (four)
10	**BCD DWord**, WLo (Lo - Hi) - WHi (Lo - Hi)	Between 0 (zero) and 99999999	4 (four)
11	**Hex DWord**, WLo (Lo - Hi) - WHi (Lo - Hi)	Between 0 (zero) and 4294967295	4 (four)

N2/B2	DATA TYPE	RANGE OF VALUES	BYTES
12	**Hex DWord**, WHi (Hi - Lo) - WLo (Hi - Lo)	Between 0 (zero) and 4294967295	4 (four)
13	**Floating point ANSI/IEEE 754 - 1985**	-	4 (four)
14	**Date and Time** in 32-bit **UTC** (*Coordinated Universal Time*) format, that is, the number of seconds since 01/01/1970	-	4 (four)
15	**Hex INT16**, (Lo - Hi)	Between -32768 and 32767	2 (two)
16	**Hex INT16**, (Hi - Lo)	Between -32768 and 32767	2 (two)
17	**Hex INT32**, WLo (Lo - Hi) - WHi (Lo - Hi)	Between -2147483648 and 2147483647	4 (four)
18	**Hex INT32**, WHi (Hi - Lo) - WLo (Hi - Lo)	Between -2147483648 and 2147483647	4 (four)
19	**String** terminated in 0 (zero)	Up to 256 characters, including the null terminator	-
20	**Struct**	-	-

NOTE

Hi: Most significant byte, **Lo**: Least significant byte, **WHi**: Most significant **Word**, **WLo**: Least significant **Word**.

BCD Word

Reading and Writing

N1/B1	Address of a device
N2/B2	0 (zero)
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a 16-bit word in **BCD** format, in which the first byte is the most significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 4 (four)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 8 (eight)
- In 4004.72R modules, the maximum number of Elements in this Tag is 16
- The range of values varies from 0 (zero) to 9999

Inverted BCD Word

Reading and Writing

N1/B1	Address of a device
N2/B2	1 (one)
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a 16-bit word in **BCD** format, in which the first byte is the least significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 4 (four)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 8 (eight)
- In 4004.72R modules, the maximum number of Elements in this Tag is 16
- The range of values varies from 0 (zero) to 9999

Inverted Hex Word

Reading and Writing

N1/B1	Address of a device
N2/B2	2 (two)
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a 16-bit word, in which the first byte is the least significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 4 (four)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 8 (eight)
- In 4004.72R modules, the maximum number of Elements in this Tag is 16
- The range of values varies from 0 (zero) to 65535

Hexadecimal Word

Reading and Writing

N1/B1	Address of a device
N2/B2	3 (three)
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a 16-bit word, in which the first byte is the most significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 4 (four)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 8 (eight)
- In 4004.72R modules, the maximum number of Elements in this Tag is 16
- The range of values varies from 0 (zero) to 65535

Byte

Reading and Writing

N1/B1	Address of a device
N2/B2	4 (four)
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes an 8-bit word.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 8 (eight)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 16
- In 4004.72R modules, the maximum number of Elements in this Tag is 32
- The range of values varies from 0 (zero) to 255

Modified Hexadecimal Word

Reading and Writing

N1/B1	Address of a device
N2/B2	5 (five)
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a 16-bit word, in which the first most significant byte is multiplied by 100 and added to the second byte.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 4 (four)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 8 (eight)
- In 4004.72R modules, the maximum number of Elements in this Tag is 16
- The range of values varies from 0 (zero) to 25755

Digital Byte

Reading and Writing

N1/B1	Address of a device
N2/B2	6 (six)
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a digital value, that is, 0 (zero) or 1 (one). When reading, if bit 0 (zero) from a byte in a device is equal to 1 (one), the returned value is 0 (zero). If bit 0 (zero) from a byte in a device is equal to 0 (zero), the returned value is equal to 1 (one). When writing, if the written value is 0 (zero), the actual value written to a device is FFh. If the written value is different from 0 (zero), the actual value written to a device is FEh.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 8 (eight)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 16
- In 4004.72R modules, the maximum number of Elements in this Tag is 32

- Possible values are 0 (zero) or 1 (one)

Negated Hexadecimal Word

Reading and Writing

N1/B1	Address of a device
N2/B2	7 (seven)
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a 16-bit word with negated bits, in which the first byte is the most significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 4 (four)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 8 (eight)
- In 4004.72R modules, the maximum number of Elements in this Tag is 16
- The range of values varies from 0 (zero) to 65535

Digital Byte with Writing of Last Bit

Reading and Writing

N1/B1	Address of a device
N2/B2	8 (eight)
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a digital value, that is, 0 (zero) or 1 (one). This tag works very similarly to a **Digital Byte Tag**, except that, when writing, this Tag only changes the last bit in the byte. To do so, this Driver performs a reading before the writing operation, masking the last bit with an appropriate value and rewriting that masked byte.

When reading, if bit 0 (zero) from a byte in a device is equal to 1 (one), the returned value is 0 (zero). If bit 0 (zero) from a byte in a device is equal to 0 (zero), the returned value is 1 (one). When writing, if the written value is 0 (zero), the least significant bit of a byte in a device is configured. If the written value is different from 0 (zero), the least significant bit of a byte in a device is zeroed.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 8 (eight)

- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 16
- In 4004.72R modules, the maximum number of Elements in this Tag is 32
- Possible values are 0 (zero) or 1 (one)

BCD DWord

Reading and Writing

N1/B1	Address of a device
N2/B2	9 (nine)
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a 32-bit word in **BCD** format, in which the first byte is the most significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 2 (two)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 4 (four)
- In 4004.72R modules, the maximum number of Elements in this Tag is 8 (eight)
- The range of values varies from 0 (zero) to 99999999

Inverted BCD DWord

Reading and Writing

N1/B1	Address of a device
N2/B2	10
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a 32-bit word in **BCD** format, in which the first byte is the least significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 2 (two)

- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 4 (four)
- In 4004.72R modules, the maximum number of Elements in this Tag is 8 (eight)
- The range of values varies from 0 (zero) to 99999999

Inverted Hexadecimal DWord

Reading and Writing

N1/B1	Address of a device
N2/B2	11
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a 32-bit word, in which the first byte is the least significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 2 (two)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 4 (four)
- In 4004.72R modules, the maximum number of Elements in this Tag is 8 (eight)
- The range of values varies from 0 (zero) to 4294967295

Hexadecimal DWord

Reading and Writing

N1/B1	Address of a device
N2/B2	12
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a 32-bit word, in which the first byte is the most significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 2 (two)

- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 4 (four)
- In 4004.72R modules, the maximum number of Elements in this Tag is 8 (eight)
- The range of values varies from 0 (zero) to 4294967295

Floating Point ANSI/IEEE 754 - 1985

Reading and Writing

N1/B1	Address of a device
N2/B2	13
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a 32-bit floating point value, according to **ANSI/IEEE 754 - 1985** standard.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 2 (two)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 4 (four)
- In 4004.72R modules, the maximum number of Elements in this Tag is 8 (eight)

Date and Time

Reading and Writing

N1/B1	Address of a device
N2/B2	14
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a date and time value from a device in 32-bit **UTC** (*Coordinated Universal Time*) format, that is, the number of seconds since 01/01/1970.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 2 (two)

- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 4 (four)
- In 4004.72R modules, the maximum number of Elements in this Tag is 8 (eight)

Inverted Hexadecimal INT16

Reading and Writing

N1/B1	Address of a device
N2/B2	15
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a signed 16-bit word, in which the first byte is the least significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 4 (four)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 8 (eight)
- In 4004.72R modules, the maximum number of Elements in this Tag is 16
- The range of values varies from -32768 to 32767

Hexadecimal INT16

Reading and Writing

N1/B1	Address of a device
N2/B2	16
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a signed 16-bit word, in which the first byte is the most significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 4 (four)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 8 (eight)

- In 4004.72R modules, the maximum number of Elements in this Tag is 16
- The range of values varies from -32768 to 32767

Inverted Hexadecimal INT32

Reading and Writing

N1/B1	Address of a device
N2/B2	17
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a signed 32-bit word, in which the first byte is the least significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 2 (two)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 4 (four)
- In 4004.72R modules, the maximum number of Elements in this Tag is 8 (eight)
- The range of values varies from -2147483648 to 2147483647

Hexadecimal INT32

Reading and Writing

N1/B1	Address of a device
N2/B2	18
N3/B3	Absolute address
N4/B4	Not used

This Tag reads and writes a signed 32-bit word, in which the first byte is the most significant one.

- In devices from MPC1200, MPC1600, MPC2002, MPC4004.01, MPC4004.02, MPC4004.04, MPC4004.09, MPC4004.11, MPC4004.12, MPC4004.11L, MPC4004.12L, and EXPERT series and all controllers from MPC2200 series, the maximum number of Elements in this Tag is 2 (two)
- In devices from MPC4004.04B, MPC4004.05B, MPC4004.06B, MPC4004.09B, MPC4004.05E, MPC4004.06E, MPC4004.09E, MPC4004.05R, MPC4004.06R, MPC4004.09R, MPC4004.05T, MPC4004.06T, and MPC4004.09T series and in module 4004.72, the maximum number of Elements in this Tag is 4 (four)
- In 4004.72R modules, the maximum number of Elements in this Tag is 8 (eight)

- The range of values varies from -2147483648 to 2147483647

String

Reading and Writing

N1/B1	Address of a device
N2/B2	19
N3/B3	Absolute address
N4/B4	Size of this **String**

This Tag reads or writes null-terminated **String** values in the specified address. It must be defined as a PLC Tag or a Block Tag with a single Element.

This Tag is not linear, that is, it cannot be grouped using **Elipse E3**, **Elipse Power**, or **Elipse Water** Superblocks feature.

The *N4* or *B4* parameter contains the size in characters of the **String** to read or write, including a null terminator. The maximum value is 256 characters.

Struct

Reading and Writing

N1/B1	Address of a device
N2/B2	20
N3/B3	Absolute address
N4/B4	Type of this structure

This Tag reads or writes values from structure elements, as configured on the **APR03** tab. The Elements of this Block Tag can have **values from different data types**.

Reading this Tag can only be performed in blocks and only accepts writing Block Elements. This Tag is not linear, that is, it cannot be grouped using **Elipse E3**, **Elipse Power**, or **Elipse Water** Superblocks feature.

The *N4* or *B4* parameter indicates the type of structure. This value references a specific structure defined on the configuration window, as described on topic **Configuration File for Struct Data Types**.

Reading Data by Event

Read-Only

N1/B1	Address of a device
N2/B2	100
N3/B3	Table number
N4/B4	Not used

This Tag is reported by events and allows reading a certain mass memory table of a device.

At each scan, or reading by scan, this Tag check in the **TABLE STATUS ADDRESS** register if there are new registers to read in the current table, defined in the *N3* parameter. If there are no registers, this Tag returns an empty list of events, that is, it does not return error nor success, and its properties remain unchanged.

If there are new registers, these ones are returned immediately as a sequence of events. For each event or register returned, this Driver fills this Tag's properties and values with data from that event and generates an **OnRead** event.

Starting with version **2.5** of this Driver, downloading events is performed by an additional thread in background. Therefore, this Tag always returns fast, because it does not wait for a complete download. So, this Tag can be configured with a very low scan value, without a negative impact on performance. This Tag's reading operation only checks for registers previously collected by the download thread. If these registers exist, they are returned immediately. If they do not exist, this Tag checks for the existence of new registers in the controller. If there are registers to read in a device, a new download request is scheduled and this Tag returns without a value, that is, an empty list. The unique download thread processes requests one by one, in the order they are scheduled. For more information, please check topic **Reading Data by Event**.

Each Block Element must correspond to a certain element on the associated table structure, as declared in the configuration file.

At each **OnRead** event of this Tag, the **Timestamp** property shows date and time of the current register read from a device.

Reading Data by Event

To read data by event from a device, users must implement and maintain control registers and an in-memory table, which must be managed and updated by a controller's application. This application must create a circular buffer, in which a data table must be kept.

This data table is formed by control addresses and registers containing a timestamp and a fixed number of data elements, according to the next table.

Format of a data table

REGISTER	TIMESTAMP	DATA 1	DATA 2	...	DATAN
1					
2					
3					
4					
...					
N					

In addition to this table, the **RECORD POINTER**, **TABLE STATUS**, and **ACQUISITION STATUS** control registers must be kept by a device's application. Please check topics **Configuration Parameters of a Table** and **Procedure to Read Records from a Table of a Device** for more information about these registers.

Configuration Parameters of a Table

RECORD POINTER

This address contains the position of the last record stored on a table. This is an index starting at 1 (one), in which 1 (one) indicates that the last record stored is at the beginning of a table, that is, **START ADDRESS**.

NOTE
Data recording must be performed in ascending order.

TABLE STATUS

Number of records stored on a table since the last update.

NOTE
The application must read the number of records indicated by this parameter. The number of records read from a table by an application must be indicated to a device by using the **ACQUISITION STATUS** control address and, when this number is equal to **TABLE STATUS**, both are zeroed by that device, indicating that the update was finished.

ACQUISITION STATUS

This address indicates to a controller the number of records read from a table in the current cycle.

NOTE
When the value of **ACQUISITION STATUS** is equal to **TABLE STATUS**, that indicates the update was finished.

REGISTERS

Ranges from 0 (zero) to the maximum number of registers.

Data Type

It must be defined for each element of a table structure, corresponding to the **Data Types Interpreted by this Driver**.

START ADDRESS

Memory address of the first data from the first record.

PACKET SIZE

Depends on the CPU used.

Previous information must be configured in a file on this Driver's configuration window. The next code shows a configuration structure to define on this window.

```
TABLE X
RECORD POINTER ADDRESS: XXXX
TABLE STATUS ADDRESS: XXXX
ACQUISITION STATUS ADDRESS: XXXX
REGISTERS: X
START ADDRESS: X
PACKET SIZE: X
STRUCT
{
  1, // WORDBCDLH
  0, // WORDBCDHL
  3, // WORDHEXAHL
...
}
```

NOTE

Values for register addresses must be provided in decimal format or in hexadecimal format with an H suffix, such as 2C00H, which corresponds to 11264.

Procedure to Read Records from a Table of a Device

This Driver reads records in descending order, that is, from the most recent to the oldest one, starting by the address indicated by **RECORD POINTER** and moving toward **START ADDRESS**. The number of records to read sequentially, starting from **RECORD POINTER**, is defined by the value of **TABLE STATUS**. This Driver must read this number of registers sequentially and then check again if this number was increased, by reading additional records starting from the new **RECORD POINTER**, if needed.

In case the value of **TABLE STATUS** is greater than the maximum size of a data packet, **PACKET SIZE**, this reading must be performed in parts. After each reading operation of a device's record blocks, this Driver must update the value of **ACQUISITION STATUS** with the current value of records already read in the current cycle.

The table implemented in a device is circular, that is, if the number of records to read is greater than the difference between **RECORD POINTER** and **START ADDRESS**, this Driver must read the records in descending order until reaching **START ADDRESS**, moving the reading toward the top of the stack, in the address resulting from the sum of **START ADDRESS** plus the value of **PACKET SIZE**.

Each time the value of **ACQUISITION STATUS** reaches the value of **TABLE STATUS**, the device's application zeroes the content of **TABLE STATUS**, which stops the current reading cycle. This Driver must proceed with the current scan of **TABLE STATUS** at each reading operation of the associated Tag, waiting to read a non-null value to start a new reading cycle.

Documentation of I/O Interfaces

This section contains the documentation of I/O Interfaces referring to the **APR03** 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.

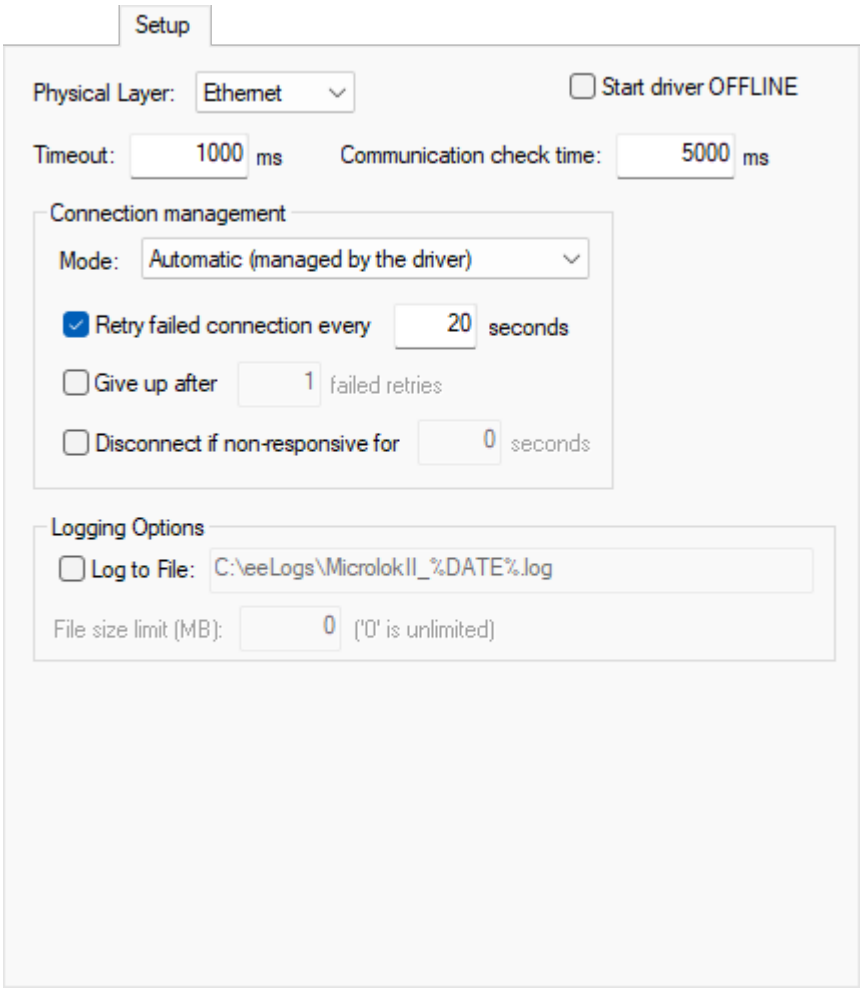
Configuration Dialog Box

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

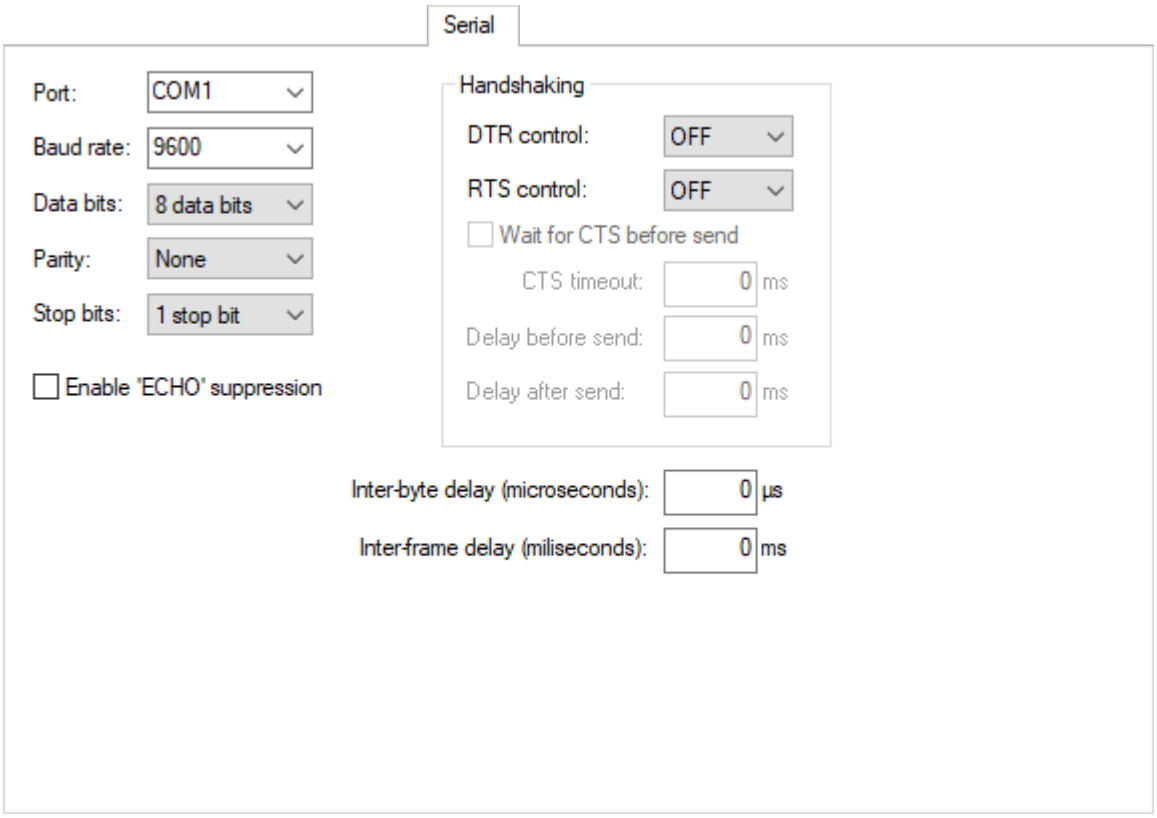
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection
Retry failed connection every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

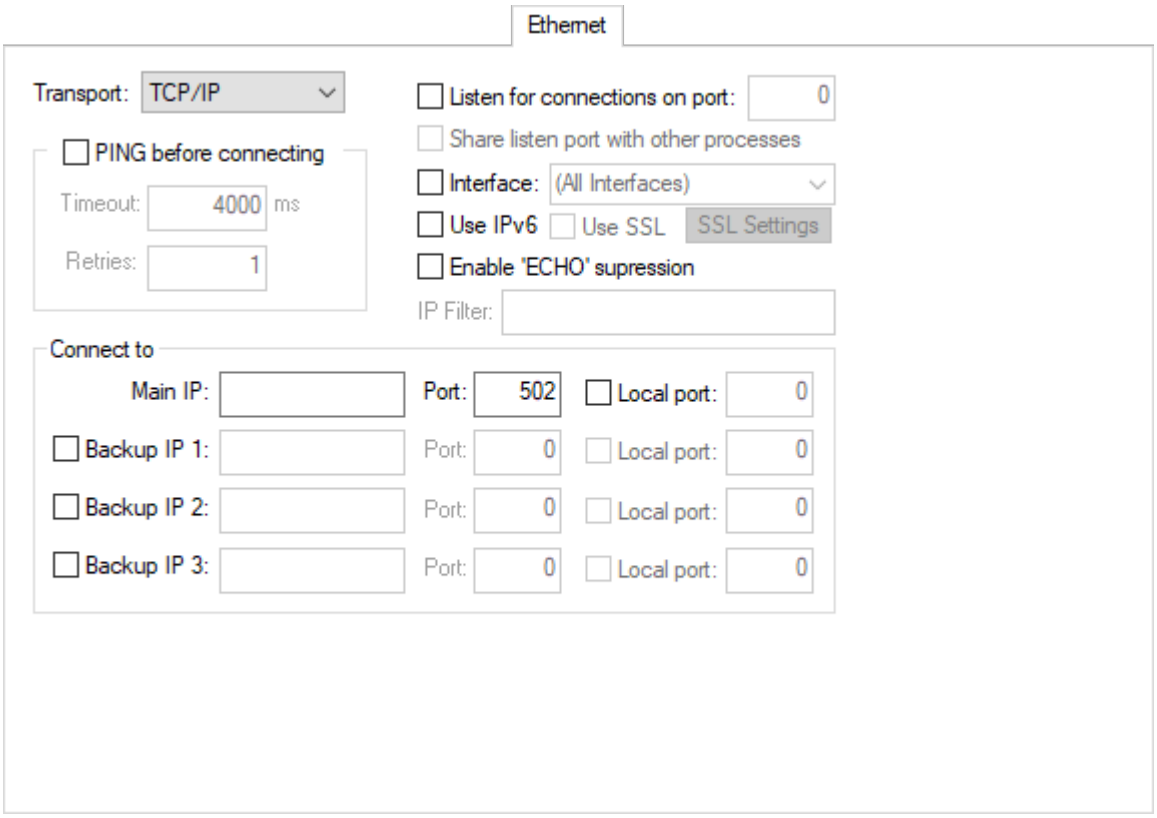
The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process, that is, it controls when data can be sent or received via serial line. Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works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

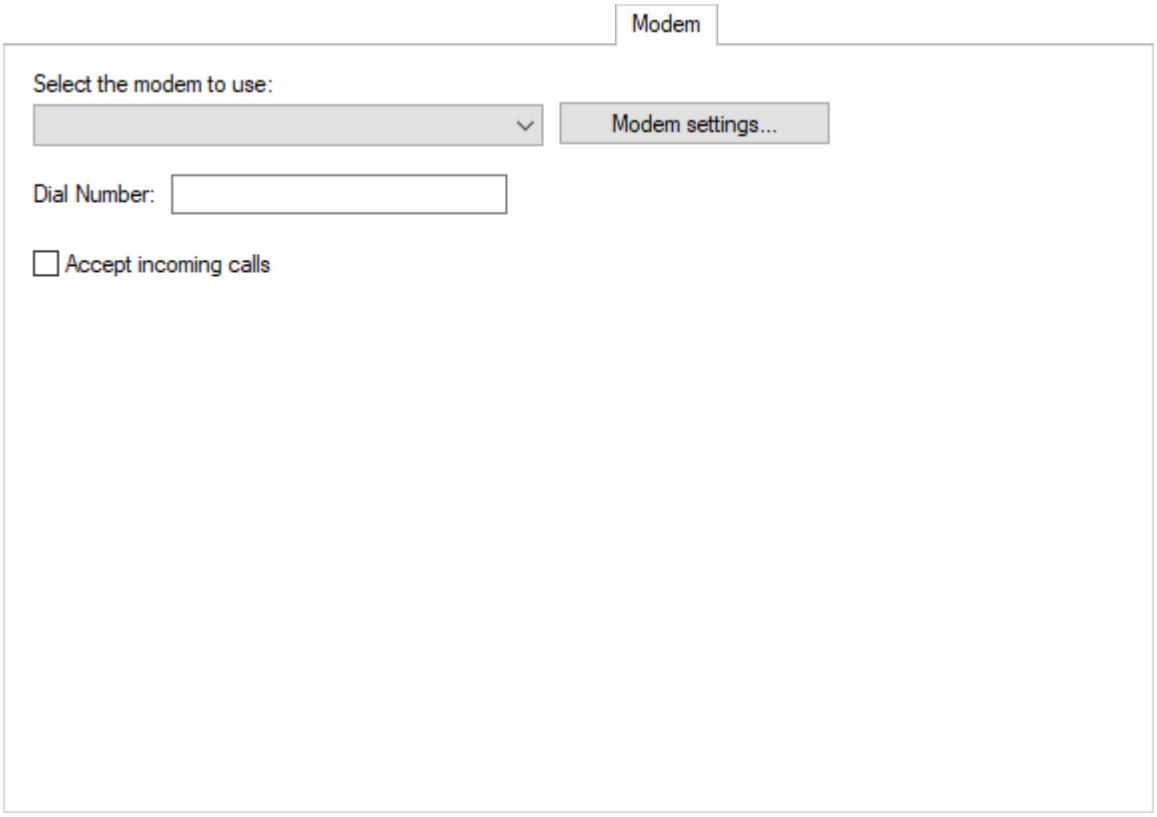
OPTION	DESCRIPTION
	of determining a successful connection before trying to open a socket with a device. The time-out of a connection with a socket can be very high. The available options are: • **Timeout**: Specify the number of milliseconds to wait for a reply from a **ping** command. Users must use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

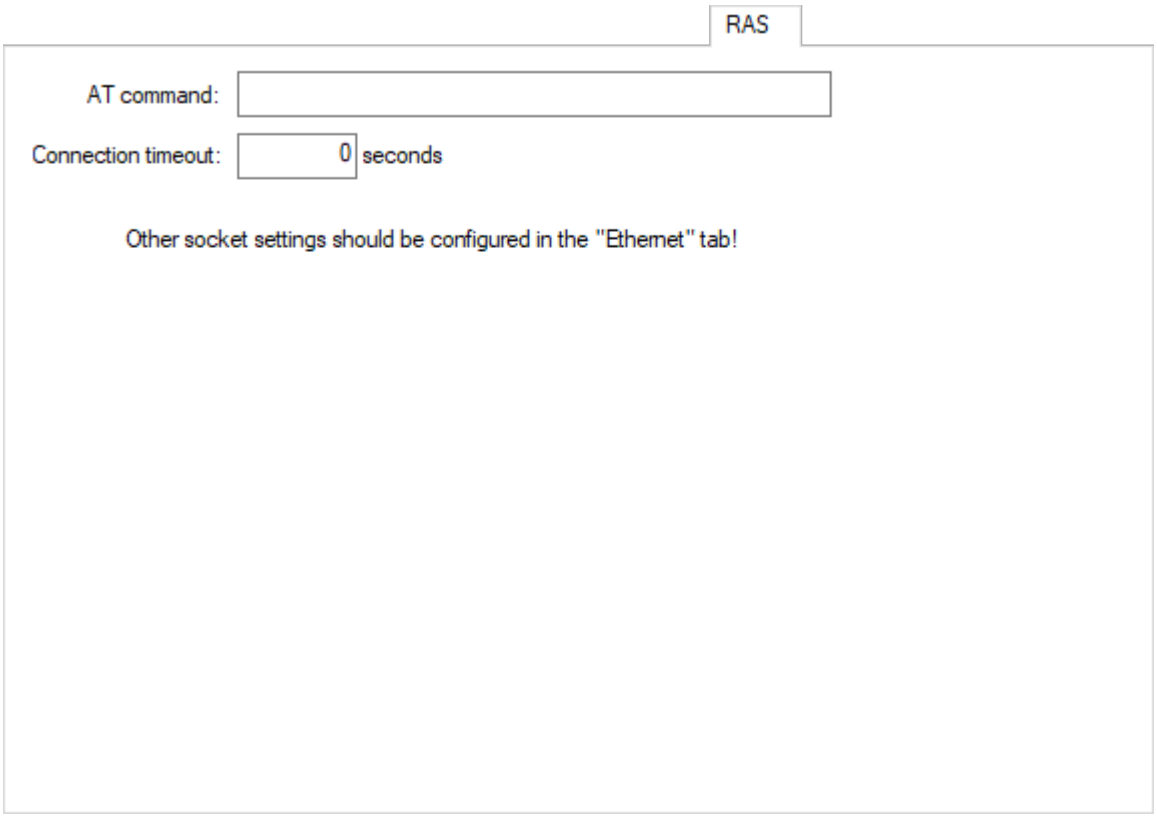
A **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

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

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

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 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

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

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

IO.Serial.Parity

Specifies a parity for the configuration of 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

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

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

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

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
3.0.2	09/01/2025	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 37940*).
3.0.1	03/16/2020	C. Mello	• Driver ported to **IOKit** library version **2.0** (*Case 27446*).
2.6.1	05/30/2011	A. Quites	• Fixed a vulnerability, which now rejects invalid frames but with a correct check byte, indicating an expected number of data bytes but with less bytes (*Case 12264*).
2.5.1	08/02/2010	A. Quites	• Fixed the way of declaring **Struct** data types in documentation (*Case 10774*).

VERSION	DATE	AUTHOR	COMMENTS
			• Data area separated to optimize and allow implementing threads to easily collect data (*Case 10626*). • Fixed an error generated by the compilation of table files from applications in versions **2.2** and **2.3**, when updated to version **2.4** (*Case 10690*). • Added a thread to allow downloading event tables in background (*Case 10625*).
2.4.1	06/08/2009	A. Quites	• Driver changed to allow downloading multiple event tables (*Case 10450*).
2.3.1	10/09/2008	A. Quites	• Fixed an error when writing **Date and Time** data types (*Case 9847*). • Implemented a detection of delayed responses (*Case 9843*).
2.2.1	07/25/2008	A. Quites	• Implemented a **Struct** data type (*Case 9257*). • Implemented Timestamp and Mass Memory Reading features (*Case 8266*). • Implemented support for **String** data types (*Case 8766*). • Fixed an error in the date and time in **UTC** format in data type **14** (*Case 9070*). • Fixed a problem preventing the use of **IOKit** library's **Physical Layer Status** Tag, which was not recognized by this Driver after the implementation of Superblocks (*Case 8816*).
2.1.1	01/17/2007	A. Quites	• Added support for Superblocks (*Case 7663*).
2.0.1	08/11/2006	A. Quites C. Mello	• Driver ported to **IOKit** library (*Case 3882*). • Implemented 32-bit and floating point data types (*Case 3943*). • Implemented a **Date and Time** data type and signed

VERSION	DATE	AUTHOR	COMMENTS
			16-bit and 32-bit data types (*Case 6457*).
1.0.1		R. Haetinger	• All releases previous to revision control, without the **IOKit** library.

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)