# Matsushita Mewtocol-COM Driver

| | |
|---|---|
| **File Name** | MewtocolCOM.dll |
| **Manufacturer** | Matsushita Electric Works, Ltd |
| **Devices** | Controllers supporting Mewtocol-COM protocol |
| **Protocol** | Mewtocol-COM |
| **Version** | 4.0.1 |
| **Last Update** | 02/13/2026 |
| **Platform** | Win32 and WinCE |
| **Dependencies** | IOKit version 1.15 or later |
| **Superblock Readings** | No |
| **Level** | 0 |

## Introduction

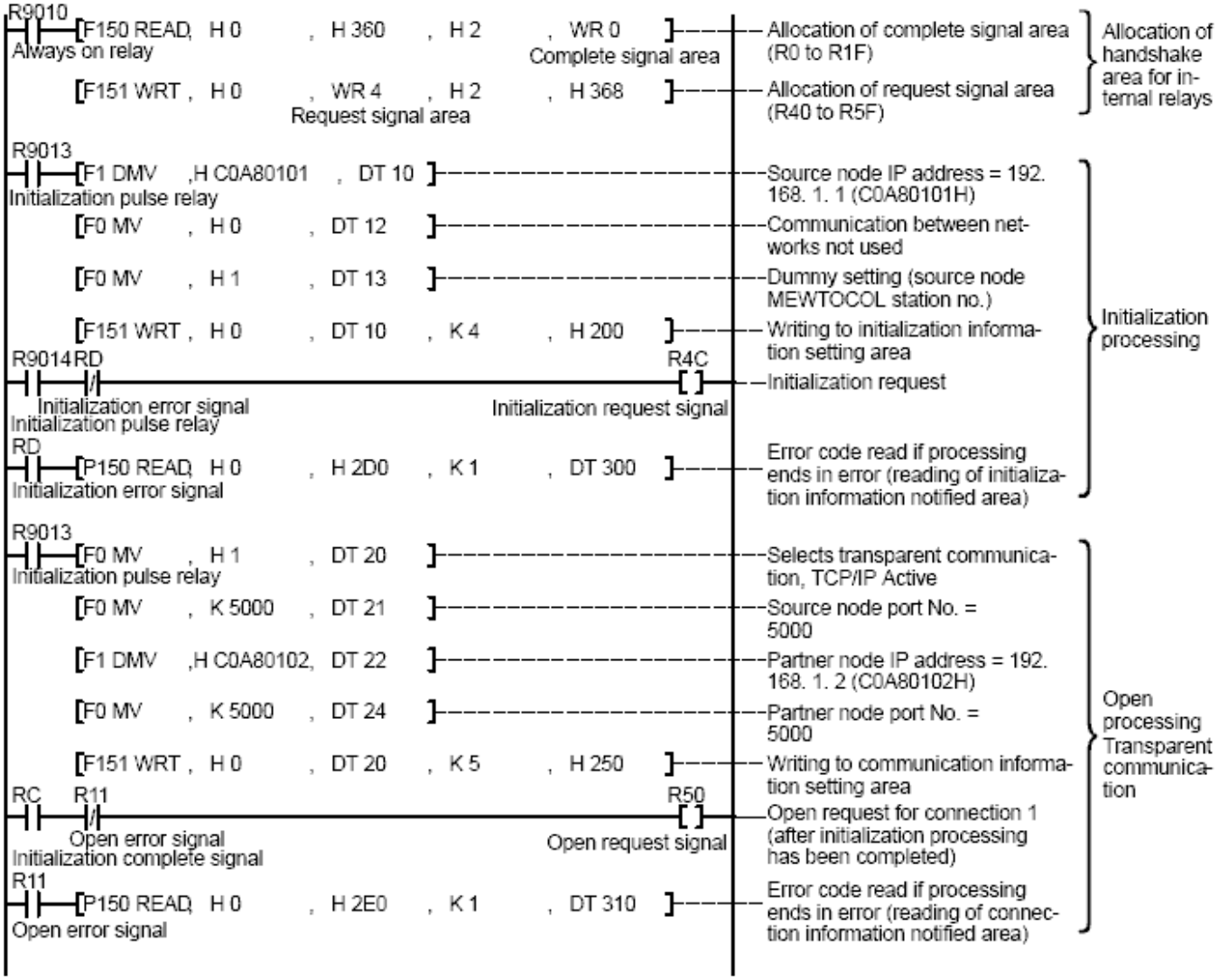
This is the Matsushita Mewtocol-COM Driver, used for communication between **Elipse Software** applications and devices supporting Matsushita Electric Works, Ltd protocol.

# Preparing a Device

This Driver allows communication using the Mewtocol-COM protocol encapsulated in any physical layer provided by the version of **IOKit** library in use.

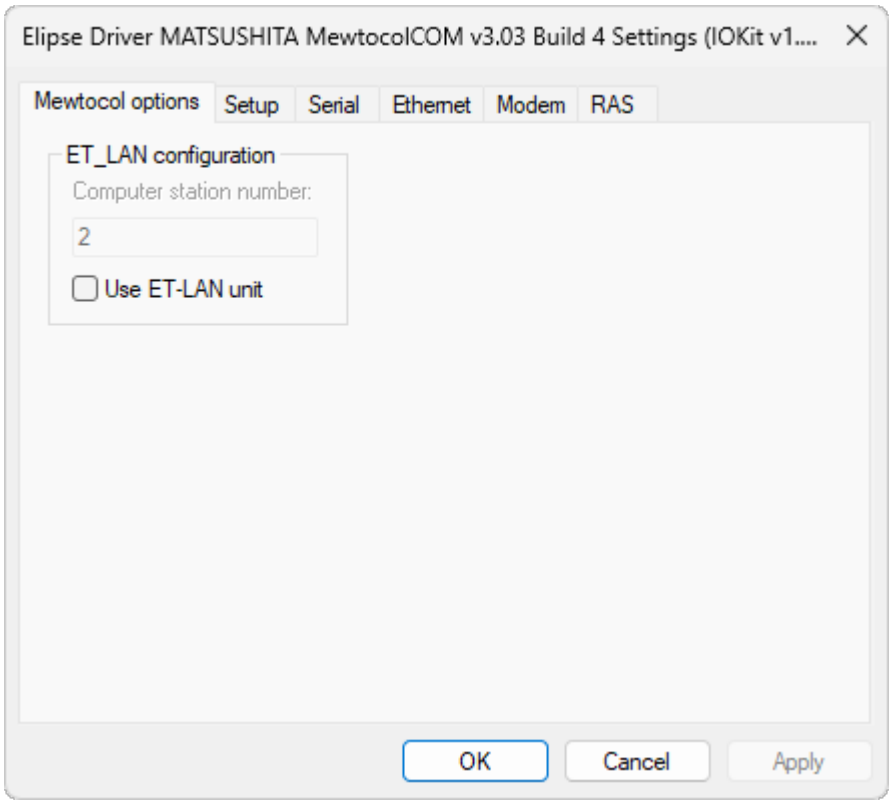
- **Serial Communication**: Users can connect a device directly to an RS-232C serial port of a computer by using a serial cable from that device. To do so, enable serial communication in a device using *System Register 412*. Also configure serial port parameters of that device using *System Register 412*, according to the configuration performed on the configuration window of this Driver. The **Terminal Code** and **Start Code** options must be left with their default values, that is, **CR** and **STX None**. For more information, please check the device's programming manual

- **TCP/IP Communication**: Users can communicate with a device in **TCP/IP** mode using the **Ethernet** module, which can be purchased separately. For more information, please check with a manufacturer. To use this module, regarding the configuration of this Driver, enable the **Use ET-LAN unit** option on the **Mewtocol options** tab of this Driver's configuration window in **Elipse E3**, **Elipse Power**, or **Elipse Water** or on the extra configuration window in **Elipse SCADA**, and specify the address that identifies this Driver on a network in the **Computer Station Number** option. The configuration of a device, on the other hand, must be performed using the programming software loaded and executed in that device. For more information, please check the programming manual and the technical manual of the **Ethernet** module for examples of configuration applications in relay language, shown on the next figure




**Example of an application for communication with an FP2-LAN module (ET-LAN unit)**

Another possible configuration, already used by some users, is using a serial communication encapsulated in TCP/IP without an **Ethernet** module. In this case, deselect the **Use ET-LAN unit** option and configure this Driver for communication in **TCP/IP** mode. To do so, use a TCP/IP to serial converter on the device side.




**Mewtocol options tab**

# Driver's [P] Configuration Parameters

| | |
|---|---|
| **P1** | Not used, leave it in 0 (zero) |
| **P2** | Not used, leave it in 0 (zero) |
| **P3** | Not used, leave it in 0 (zero) |
| **P4** | Not used, leave it in 0 (zero) |

This Driver does not use **[P]** configuration parameters. All settings regarding communication must be performed on the configuration window. For more information about the configuration options of **IOKit** library, please check topic **Documentation of I/O Interfaces**.

# Tags Reference

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

# [N/B] Parameters for Tag Addressing

This Driver does not differentiate between Block and PLC Tags, that is, PLC Tags are always handled as Block Tags with a single Element.

The Tags of this Driver allow reading and writing operands with **Bit**, **Word** (16 bits), floating point (32 bits), and **DWord** (32 bits) data types, as well as reading status information from a device. Please check the description of each Tag for more details about their functionality and configuration.

## X Contacts in Bit or Word

### Read-Only

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1000 |
| **N3 or B3** | Number of the initial contact, multiple of 16 |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of a block of 16 external input **X** contacts. It is mandatory using blocks multiples of 16, in which each Element represents a contact. Reading is performed with **Word** data types using the **RCCX** command of Mewtocol-COM protocol, therefore the address of data, the *N3* parameter, must be a value multiple of 16, that is, the beginning of a **Word** value. In case of reading a single Element, the **RCSX** command of Mewtocol-COM protocol is used.

**NOTE**

When using Block Tags, the maximum number of Elements is 96.

# Y Contacts in Bit or Word

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1001 |
| **N3 or B3** | Number of the initial contact, multiple of 16 |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of a block of 16 external output **Y** contacts. It is mandatory using blocks multiples of 16, in which each Element represents a contact. Reading or writing is performed with **Word** data types using the **RCCY** or **WCCY** commands, respectively, of Mewtocol-COM protocol, therefore the address of data, the *N3* parameter, must be a value multiple of 16, that is, the beginning of a **Word** value. In case of reading or writing a single Element, the **RCSY** or **WCSY** commands, respectively, of Mewtocol-COM protocol is used.

**NOTE**

When using Block Tags, the maximum number of Elements is 96.

# R Contacts in Bit or Word

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1002 |
| **N3 or B3** | Number of the initial contact, multiple of 16 |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of a block of 16 internal relay **R** contacts. It is mandatory using blocks multiples of 16, in which each Element represents a contact. Reading or writing is performed with **Word** data types using the **RCCR** or **WCCR** commands, respectively, of Mewtocol-COM protocol, therefore the address of data, the *N3* parameter, must be a value multiple of 16, that is, the beginning of a **Word** value. In case of reading or writing a single Element, the **RCSR** or **WCSR** commands, respectively, of Mewtocol-COM protocol is used.

**NOTE**

When using Block Tags, the maximum number of Elements is 96.

# L Contacts in Bit or Word

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1003 |
| **N3 or B3** | Number of the initial contact, multiple of 16 |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of a block of 16 link relay **L** contacts. It is mandatory using blocks multiples of 16, in which each Element represents a contact. Reading or writing is performed with **Word** data types using the **RCCL** or **WCCL** commands, respectively, of Mewtocol-COM protocol, therefore the address of data, the *N3* parameter, must be a value multiple of 16, that is, the beginning of a **Word** value. In case of reading or writing a single Element, the **RCSL** or **WCSL** commands, respectively, of Mewtocol-COM protocol is used.

**NOTE**

When using Block Tags, the maximum number of Elements is 96.

# T Contacts in Bit or Word

### Read-Only

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1004 |
| **N3 or B3** | Number of the initial contact, multiple of 16 |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of a block of 16 **T** (*Timer*) contacts. It is mandatory using blocks multiples of 16, in which each Element represents a contact. Reading is performed with **Word** data types using the **RCCT** command of Mewtocol-COM protocol, therefore the address of data, the *N3* parameter, must be a value multiple of 16, that is, the beginning of a **Word** value. In case of reading a single Element, the **RCST** command of Mewtocol-COM protocol is used.

**NOTE**

When using Block Tags, the maximum number of Elements is 96.

# C Contacts in Bit or Word

### Read-Only

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1005 |
| **N3 or B3** | Number of the initial contact, multiple of 16 |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of a block of 16 **C** (*Counter*) contacts. It is mandatory using blocks multiples of 16, in which each Element represents a contact. Reading is performed with **Word** data types using the **RCCC** command of Mewtocol-COM protocol, therefore the address of data, the *N3* parameter, must be a value multiple of 16, that is, the beginning of a **Word** value. In case of reading a single Element, the **RCSX** command of Mewtocol-COM protocol is used.

**NOTE**

When using Block Tags, the maximum number of Elements is 96.

# DT Data Registers

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1006 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. For more information, please check the next **table** |

This Tag reads or writes from or to **DT** data registers.

**Values for data types of the N4 or B4 parameter**

| VALUE | DESCRIPTION |
|---|---|
| **0** | Unsigned 16-bit integer, between 0 (zero) and 65536 |
| **1** | Signed 16-bit integer, between -32767 and 32767 |
| **2** | 32-bit floating point, between ±1.175494 × $10^{-38}$ and 3.402823 × $10^{38}$ |
| **3** | Unsigned 32-bit integer, between 0 (zero) and 4294967295 |
| **4** | Signed 32-bit integer, between -2147483648 and 2147483647 |
| **1000** | **String** with a variable size of 4 (four) characters for each 16-bit **Word** |
| **2000** | **String** in hexadecimal with a variable size of 2 (two) characters for each 16-bit **Word** |

**NOTES**

- For 32-bit data types, 2 (two) registers are used to represent a single number for each Element of a Block Tag, in which the *N3* parameter is the number of this Element. In these cases the address of data, the *N3* parameter, must be defined as the first value with a **Word** data type of the number to read or write. For signed or unsigned integer data types, or 16 bits, reading or writing is performed in a simple way, in which each Element of a Block Tag represents data in a device.
- For reading data in text format, the values 1000 and 2000, use only PLC Tags. When performing a reading of data in text format, indicate in the *N4* parameter the number of 16-bit *n* **Words** to receive, specifying the value 1000 + *n* for **Strings** or 2000 + *n* for **Strings** in hexadecimal. For example, the value 1003 indicates a reading of 3 (three) **Word** values with 12 characters and the value 2003 indicates a reading of 3 (three) **Word** values with 6 (six) characters. For writing data in text format, there is no need to inform the number of **Words** in the *N4* parameter, therefore this parameter can keep the values 1000 for **Strings** and 2000 for **Strings** in hexadecimal.

# LD Link Registers

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1007 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. For more information, please check the next **table** |

This Tag reads or writes from or to **LD** link data registers.

**Values for data types of the N4 or B4 parameter**

| VALOR | DESCRIÇÃO |
|---|---|
| **0** | Unsigned 16-bit integer, between 0 (zero) and 65536 |
| **1** | Signed 16-bit integer, between -32767 and 32767 |
| **2** | 32-bit floating point, between ±1.175494 × $10^{-38}$ and 3.402823 × $10^{38}$ |
| **3** | Unsigned 32-bit integer, between 0 (zero) and 4294967295 |
| **4** | Signed 32-bit integer, between -2147483648 and 2147483647 |

**NOTE**

For 32-bit data types, 2 (two) registers are used to represent a single number for each Element of a Block Tag, in which the *N3* parameter is the number of this Element. In these cases the address of data, the *N3* parameter, must be defined as the first value with a **Word** data type of the number to read or write. For signed or unsigned integer data types, or 16 bits, reading or writing is performed in a simple way, in which each Element of a Block Tag represents data in a device.

# FL File Registers

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1008 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. For more information, please check the next **table** |

This Tag reads or writes from or to **FL** file registers.

**Values for data types of the N4 or B4 parameter**

| VALOR | DESCRIÇÃO |
|---|---|
| **0** | Unsigned 16-bit integer, between 0 (zero) and 65536 |
| **1** | Signed 16-bit integer, between -32767 and 32767 |
| **2** | 32-bit floating point, between ±1.175494 × $10^{-38}$ and 3.402823 × $10^{38}$ |
| **3** | Unsigned 32-bit integer, between 0 (zero) and 4294967295 |
| **4** | Signed 32-bit integer, between -2147483648 and 2147483647 |

**NOTE**

For 32-bit data types, 2 (two) registers are used to represent a single number for each Element of a Block Tag, in which the *N3* parameter is the number of this Element. In these cases the address of data, the *N3* parameter, must be defined as the first value with a **Word** data type of the number to read or write. For signed or unsigned integer data types, or 16 bits, reading or writing is performed in a simple way, in which each Element of a Block Tag represents data in a device.

# X Index Registers

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1009 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. For more information, please check the next **table** |

This Tag reads or writes from or to **X** (IX) index registers referring to I0 data. The maximum number of Elements of this Block Tag is 1 (one).

**Values for data types of the N4 or B4 parameter**

| VALOR | DESCRIÇÃO |
|---|---|
| **0** | Unsigned 16-bit integer, between 0 (zero) and 65536 |
| **1** | Signed 16-bit integer, between -32767 and 32767 |

# Y Index Registers

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1010 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. For more information, please check the next **table** |

This Tag reads or writes from or to **Y** (IY) index registers referring to I1 data. The maximum number of Elements of this Block Tag is 1 (one).

**Values for data types of the N4 or B4 parameter**

| VALOR | DESCRIÇÃO |
|---|---|
| **0** | Unsigned 16-bit integer, between 0 (zero) and 65536 |
| **1** | Signed 16-bit integer, between -32767 and 32767 |

# Configured Value for Timer or Counter

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1011 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. For more information, please check the next **table** |

This Tag reads or writes values referring to the timer or counter of a device in the **Set Value** area.

**Values for data types of the N4 or B4 parameter**

| VALOR | DESCRIÇÃO |
|---|---|
| **0** | Unsigned 16-bit integer, between 0 (zero) and 65536 |
| **1** | Signed 16-bit integer, between -32767 and 32767 |
| **2** | 32-bit floating point, between ±1.175494 × $10^{-38}$ and 3.402823 × $10^{38}$ |
| **3** | Unsigned 32-bit integer, between 0 (zero) and 4294967295 |
| **4** | Signed 32-bit integer, between -2147483648 and 2147483647 |

**NOTE**

For 32-bit data types, 2 (two) registers are used to represent a single number for each Element of a Block Tag, in which the *N3* parameter is the number of this Element. In these cases the address of data, the *N3* parameter, must be defined as the first value with a **Word** data type of the number to read or write. For signed or unsigned integer data types, or 16 bits, reading or writing is performed in a simple way, in which each Element of a Block Tag represents data in a device.

# Elapsed Value for Timer or Counter

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1012 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. For more information, please check the next **table** |

This Tag reads or writes 16-bit integers referring to the current elapsed value of the timer or counter of a device in the **Elapsed Value** area.

**Values for data types of the N4 or B4 parameter**

| VALOR | DESCRIÇÃO |
|---|---|
| **0** | Unsigned 16-bit integer, between 0 (zero) and 65536 |
| **1** | Signed 16-bit integer, between -32767 and 32767 |
| **2** | 32-bit floating point, between $\pm 1.175494 \times 10^{-38}$ and $3.402823 \times 10^{38}$ |
| **3** | Unsigned 32-bit integer, between 0 (zero) and 4294967295 |
| **4** | Signed 32-bit integer, between -2147483648 and 2147483647 |

**NOTE**

For 32-bit data types, 2 (two) registers are used to represent a single number for each Element of a Block Tag, in which the *N3* parameter is the number of this Element. In these cases the address of data, the *N3* parameter, must be defined as the first value with a **Word** data type of the number to read or write. For signed or unsigned integer data types, or 16 bits, reading or writing is performed in a simple way, in which each Element of a Block Tag represents data in a device.

# System Registers

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1013 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. For more information, please check the next **table** |

This Tag reads or writes 16-bit integer values of system registers of a device.

**Values for data types of the N4 or B4 parameter**

| VALOR | DESCRIÇÃO |
|---|---|
| **0** | Unsigned 16-bit integer, between 0 (zero) and 65536 |
| **1** | Signed 16-bit integer, between -32767 and 32767 |
| **2** | 32-bit floating point, between $\pm 1.175494 \times 10^{-38}$ and $3.402823 \times 10^{38}$ |
| **3** | Unsigned 32-bit integer, between 0 (zero) and 4294967295 |
| **4** | Signed 32-bit integer, between -2147483648 and 2147483647 |

**NOTE**

For 32-bit data types, 2 (two) registers are used to represent a single number for each Element of a Block Tag, in which the *N3* parameter is the number of this Element. In these cases the address of data, the *N3* parameter, must be defined as the first value with a **Word** data type of the number to read or write. For signed or unsigned integer data types, or 16 bits, reading or writing is performed in a simple way, in which each Element of a Block Tag represents data in a device.

# Status of a Device

### Read-Only

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1014 |
| **N3 or B3** | Address of data. For more information, please check the next **table** |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads status information from a device. Reading this information is performed in blocks using the **RT** command of Mewtocol-COM protocol. The address indicated in the *N3* or *B3* parameter defines which information is extracted from that block. Although this information can be read using PLC Tags, it is recommended using Block Tags, as information is read from a device only in blocks. In case of PLC Tags, this Driver must read a full status block and then return only the

requested information to an application. Preferably, create a Block Tag with 21 Elements and the *B3* parameter equal to 0 (zero) so that readings follow the configuration of this Tag's scan.

**Possible values for the N3 or B3 parameter**

| ADDRESS | DESCRIPTION | VALUE |
|---|---|---|
| **0** | Model of a device | Possible values are **4**: C16 or **5**: C24, C40 |
| **1** | Version of a device | |
| **2** | Capacity of a program | |
| **3** | Remote | Possible value is **1**: Remote |
| **4** | Message. F149 (MSG) or P149 (PMSG) | Possible value is **1**: Message enabled |
| **5** | Execution mode | Possible values are **1**: Programming online or **0**: Normal execution |
| **6** | External output | Possible values are **0**: No output for external device or **1**: With output for external device |
| **7** | Execution of a BRK command | Possible values are **0**: Invalid **BRK** command or **1**: Execution of **BRK** command enabled |
| **8** | Execution of a BRK command | Possible values are **1**: When the **BRK/1** command is executed or **0**: Normal execution |
| **9** | Test mode | Possible values are **1**: Test mode or **0**: Normal operation |
| **10** | Operation mode | Possible values are **0**: RUN or **1**: PROG |
| **11** | Flag for operation error | Possible values are **1**: With error or **0**: Without error |
| **12** | Battery error | Possible values are **1**: Turned on or **0**: Turned off |
| **13** | Battery error | Possible values are **1**: With error or **0**: Without error |
| **14** | Input and output check error | Possible values are **1**: With error or **0**: Without error |
| **15** | Error in the Intelligent Unit | Possible values are **1**: With error or **0**: Without error |
| **16** | Detection of a burning fuse | Possible values are **1**: Burning fuse detected or **0**: Burning fuse not detected |
| **17** | Detection of an interruption | Possible values are **1**: Interruption detected or **0**: Interruption not detected |
| **18** | Auto-diagnostic | Possible values are **1**: Error detected (please check the error code in the address **20**) or **0**: Error not detected |
| **19** | Status of Link network | Possible values are **1**: Active or **0**: Disabled |

| ADDRESS | DESCRIPTION | VALUE |
|---|---|---|
| **20** | Number of auto-diagnostic error | |

# Operation Mode

### Somente Escrita

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1015 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag writes to a device the operation mode using the RM command of Mewtocol-COM protocol. This command only works when a device is in remote operation mode. The available operation modes are **0**: **Run** mode or **1**: **Program** mode.

# DT Data Registers + 50000

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1056 |
| **N3 or B3** | Address of data minus 50000. For example, to access the address 98500, use this parameter with the value 48500 |
| **N4 or B4** | Data type. For more information, please check the next **table** |

This Tag reads or writes from or to DT data registers, just like the **DT Data Registers** Tag. The difference is the *B3* parameter, which adds 50000 to the value provided. This Tag must be used when a data address exceeds the limit of the *N3* parameter, which is 65535.

**Values for data types of the N4 or B4 parameter**

| VALUE | DESCRIPTION |
|---|---|
| **0** | Unsigned 16-bit integer, between 0 (zero) and 65536 |
| **1** | Signed 16-bit integer, between -32767 and 32767 |
| **2** | 32-bit floating point, between $\pm 1.175494 \times 10^{-38}$ and $3.402823 \times 10^{38}$ |
| **3** | Unsigned 32-bit integer, between 0 (zero) and 4294967295 |
| **4** | Signed 32-bit integer, between -2147483648 and 2147483647 |

| VALUE | DESCRIPTION |
|---|---|
| **1000** | **String** with a variable size of 4 (four) characters for each 16-bit **Word** |
| **2000** | **String** in hexadecimal with a variable size of 2 (two) characters for each 16-bit **Word** |

# X Contacts in Word

### Read-Only

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1100 |
| **N3 or B3** | Number of a **Word** value (16 bits) |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of a 16-bit **Word** value of external input **X** contacts. In this Tag, these contacts are read in a single Element. Each Element represents a **Word** value and the *N3* or *B3* parameter is the number of a **Word** value to read. For example, the *N3* or *B3* parameter equal to 0 (zero) corresponds to contacts from 0 (zero) to 15. To read these contacts, consider each bit of a **Word** value as the value of each contact, from the least significant bit to the most significant bit. A reading is performed using the **RCCX** command of Mewtocol-COM protocol.

# Y Contacts in Word

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1101 |
| **N3 or B3** | Number of a **Word** value (16 bits) |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads or writes the status, on or off, of a 16-bit **Word** value of external output **Y** contacts. In this Tag, these contacts are read or written in a single Element. Each Element represents a **Word** value and the *N3* or *B3* parameter is the number of a **Word** value to read or write. For example, the *N3* or *B3* parameter equal to 0 (zero) corresponds to contacts from 0 (zero) to 15. To read or write these contacts, consider each bit of a **Word** value as the value of each contact, from the least significant bit to the most significant bit. Reading or writing is performed using the **RCCY** or **WCCY** commands, respectively, of Mewtocol-COM protocol.

# R Contacts in Word

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1102 |
| **N3 or B3** | Number of a **Word** value (16 bits) |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads or writes the status, on or off, of a 16-bit **Word** value of internal relay **R** contacts. In this Tag, these contacts are read or written in a single Element. Each Element represents a **Word** value and the *N3* or *B3* parameter is the number of a **Word** value to read or write. For example, the *N3* or *B3* parameter equal to 0 (zero) corresponds to contacts from 0 (zero) to 15. To read or write these contacts, consider each bit of a **Word** value as the value of each contact, from the least significant bit to the most significant bit. A reading is performed using the **RCCR** or **WCCR** commands, respectively, of Mewtocol-COM protocol.

# L Contacts in Word

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1103 |
| **N3 or B3** | Number of a **Word** value (16 bits) |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads or writes the status, on or off, of a 16-bit **Word** value of link relay **L** contacts. In this Tag, these contacts are read or written in a single Element. Each Element represents a **Word** value and the *N3* or *B3* parameter is the number of a **Word** value to read or write. For example, the *N3* or *B3* parameter equal to 0 (zero) corresponds to contacts from 0 (zero) to 15. To read or write these contacts, consider each bit of a **Word** value as the value of each contact, from the least significant bit to the most significant bit. Reading or writing is performed using the **RCCL** or **WCCL** commands, respectively, of Mewtocol-COM protocol.

# T Contacts in Word

### Read-Only

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1104 |
| **N3 or B3** | Number of a **Word** value (16 bits) |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of a 16-bit **Word** value of **T** (*Timer*) contacts. In this Tag, these contacts are read in a single Element. Each Element represents a **Word** value and the *N3* or *B3* parameter is the number of a **Word** value to read. For example, the *N3* or *B3* parameter equal to 0 (zero) corresponds to contacts from 0 (zero) to 15. To read these contacts, consider each bit of a **Word** value as the value of each contact, from the least significant bit to the most significant bit. A reading is performed using the **RCCT** command of Mewtocol-COM protocol.

# C Contacts in Word

### Read-Only

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1105 |
| **N3 or B3** | Number of a **Word** value (16 bits) |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of a 16-bit **Word** value of **C** (*Counter*) contacts. In this Tag, these contacts are read in a single Element. Each Element represents a **Word** value and the *N3* or *B3* parameter is the number of a **Word** value to read. For example, the *N3* or *B3* parameter equal to 0 (zero) corresponds to contacts from 0 (zero) to 15. To read these contacts, consider each bit of a **Word** value as the value of each contact, from the least significant bit to the most significant bit. A reading is performed using the **RCCC** command of Mewtocol-COM protocol.

# X Contacts in Bit

### Read-Only

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1200 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of one or more external input **X** contacts. This Tag allows reading any contact address, but there is a limitation of 8 (eight) Elements and each Element corresponds to 1 (one) contact. A reading is performed using the **RCPX** command of Mewtocol-COM protocol.

# Y Contacts in Bit

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1201 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of one or more external output **Y** contacts. This Tag allows reading or writing any contact address, but there is a limitation of 8 (eight) Elements and each Element corresponds to 1 (one) contact. Reading or writing is performed using the **RCPY** or **WCPY** commands, respectively, of Mewtocol-COM protocol.

# R Contacts in Bit

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1202 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of one or more internal relay **R** contacts. This Tag allows reading or writing any contact address, but there is a limitation of 8 (eight) Elements and each Element corresponds to 1 (one) contact. Reading or writing is performed using the **RCPR** or **WCPR** commands, respectively, of Mewtocol-COM protocol.

# L Contacts in Bit

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1203 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of link relay **L** contacts. This Tag allows reading or writing any contact address, but there is a limitation of 8 (eight) Elements and each Element corresponds to 1 (one) contact. Reading or writing is performed using the **RCPL** or **WCPL** commands, respectively, of Mewtocol-COM protocol.

## T Contacts in Bit

**Read-Only**

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1204 |
| **N3 or B3** | Not used, leave it in 0 (zero) |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of **T** (*Timer*) contacts. This Tag allows reading any contact address, but there is a limitation of 8 (eight) Elements and each Element corresponds to 1 (one) contact. A reading is performed using the **RCPT** command of Mewtocol-COM protocol.

## C Contacts in Bit

**Read-Only**

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 or 255 for writing data in **Broadcast** mode |
| **N2 or B2** | 1205 |
| **N3 or B3** | Not used, leave it in 0 (zero) |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads the status, on or off, of **C** (*Counter*) contacts. This Tag allows reading any contact address, but there is a limitation of 8 (eight) Elements and each Element corresponds to 1 (one) contact. A reading is performed using the **RCPC** command of Mewtocol-COM protocol.

# Reading Variables in Batches

This Driver allows reading history data blocks stored in batches in memory. A device stores *N* samples of *M* variables from a batch in a memory area and, at the end of that batch, increments a *BN* variable and configures an *I* variable with the interval between samples, in milliseconds. A *Flag* variable is configured with the value 1 (one) whenever a device is performing a sampling and configured with the value 0 (zero) whenever a batch is ready for reading. The memory map shown next corresponds to the variables *N* equal to 100 and *M* equal to 8 (eight). All variables are normal registers of a device, that is, integers with 2 (two) bytes.

```
000: Flag (1: Batch is not ready or 0: Batch is ready)
001: BN (Batch number)
002: I
003: M
004: N
First sample
005: V1, V2, V3, V4, V5, V6, V7, V8
Second sample
013: V1, V2, V3, V4, V5, V6, V7, V8
...
...
...
Last sample
797: V1, V2, V3, V4, V5, V6, V7, V8
```

Reading and monitoring a device's history is performed using a Tag with the *B2* parameter equal to 100, according to the next table. The number of Elements of this Block Tag is equal to the number of variables read in each register. This Tag only returns data if there are batches to read and if the *Flag* variable is configured as 0 (zero). Batches already in memory at the moment of the first reading of this Tag, or the first scan after initializing an application, are considered as already read and only new batches are read.

| | |
|---|---|
| **B1** | Address of a device in the network, between 1 (one) and 63 |
| **B2** | 100 |
| **B3** | Starting address of a data block |
| **B4** | This parameter must be used only for **Word** data types. Possible values are **0**: Unsigned integer between 0 (zero) and 65536 or **1**: Signed integer between -32767 and 32767. For other data types this parameter is not used and can be left in 0 (zero) |

At each scan of this Tag, this Driver must monitor the *BN* variable (batch number) and, if it detects a change in its value, it must check whether the *Flag* variable is configured with the value 0 (zero). If true, it must transfer the history of this batch to the application, considering as the starting time the current time minus the result of (*N* × *I*) - 1, in milliseconds, and add *I* milliseconds for each new sample.

The values read from each batch are returned by this Driver to an application as a list of events, generating a succession of **OnRead** events of this Tag. For each **OnRead** event, the Elements of this Tag contain the values referring to a specific register from a batch. Users can use a script in these events to execute a **WriteRecord** method of the linked Historic object, storing the blocks read one by one.

# Appendix I

This section contains information about deprecated configuration parameters of this Driver.

# Deprecated [N/B] Parameters for Tag Addressing

The next topics contain the meaning of Tags used up to version **2.1** of this Driver. These Tags still have the same functionality but they are deprecated for the current version. It is recommended using the Tags described on topic **[N/B] Parameters for Tag Addressing**.

## X Contacts

### Read-Only

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 0 (zero) |
| **N3 or B3** | Address of data, in hexadecimal |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads external input **X** contacts.

# Y Contacts

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 1 (one) |
| **N3 or B3** | Address of data, in hexadecimal |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads or writes 1 (one) digit, or bit, from or to external output **Y** contacts.

# R Contacts

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 2 (two) |
| **N3 or B3** | Address of data, in hexadecimal |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads or writes from or to internal relay **R** contacts.

# T Contacts

### Read-Only

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 3 (three) |
| **N3 or B3** | Address of data, in hexadecimal |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads **T** (*Timer*) contacts.

# C Contacts

### Read-Only

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 4 (four) |
| **N3 or B3** | Address of data, in hexadecimal |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads **C** (*Counter*) contacts.

# Data Registers

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 5 (five) |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. Possible values are **0**: Unsigned integer between 0 (zero) and 65536 or **1**: Signed integer between -32767 and 32767 |

This Tag reads of writes 16-bit integer values of **DT** data registers.

# X Index Registers

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 6 (six) |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. Possible values are **0**: Unsigned integer between 0 (zero) and 65536 or **1**: Signed integer between -32767 and 32767 |

This Tag reads or writes 16-bit integer values of **X** (IX) index registers referring to I0 data.

# Y Index Registers

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 7 (seven) |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. Possible values are **0**: Unsigned integer between 0 (zero) and 65536 or **1**: Signed integer between -32767 and 32767 |

This Tag reads or writes 16-bit integer values of **Y** (IY) index registers referring to I1 data.

# Configured Value for Timer or Counter

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 8 (eight) |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. Possible values are **0**: Unsigned integer between 0 (zero) and 65536 or **1**: Signed integer between -32767 and 32767 |

This Tag reads or writes 16-bit integer values referring to the counter or timer of a device in the **Set Value** area. For Block Tags, the maximum number of Elements is 24.

# Elapsed Value for Timer or Counter

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 9 (nine) |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. Possible values are **0**: Unsigned integer between 0 (zero) and 65536 or **1**: Signed integer between -32767 and 32767 |

This Tag reads or writes 16-bit integer values referring to the current elapsed value of the timer or counter of a device in the **Elapsed Value** area. For Block Tags, the maximum number of Elements is 24.

# System Registers

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 10 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Data type. Possible values are **0**: Unsigned integer between 0 (zero) and 65536 or **1**: Signed integer between -32767 and 32767 |

This Tag reads or writes 16-bit integer values from a device's system registers.

# Status of a Device

### Reading or Writing

| | |
|---|---|
| **N1 or B1** | Address of a device in the network, between 1 (one) and 63 |
| **N2 or B2** | 11 |
| **N3 or B3** | Address of data |
| **N4 or B4** | Not used, leave it in 0 (zero) |

This Tag reads status information from a device. Reading this information is performed in blocks using the **RT** command of Mewtocol-COM protocol. The address indicated in the *N3* or *B3* parameter defines which information is extracted from that block. Although this information can be read by using a PLC Tag, it is recommended using Block Tags, as information is read from a device only in blocks. In case of PLC Tags, this Driver must read a full status block and then return only the information requested by an application. Preferably, create a Block Tag with 21 Elements and the *B3* parameter equal to 0 (zero) so that readings follow the scan configuration of this Tag. Please check table **Possible values for the N3 or B3 parameter** for a description of each Element or address.

# Documentation of I/O Interfaces

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

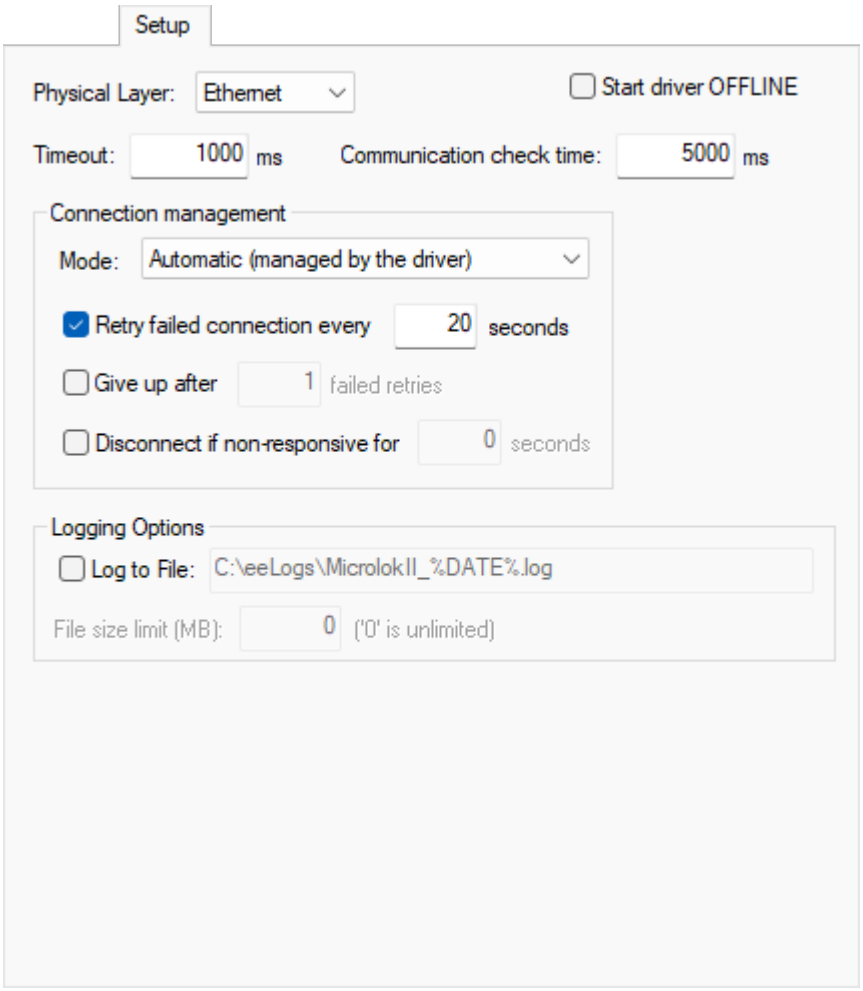
## Configuration Dialog Box

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**

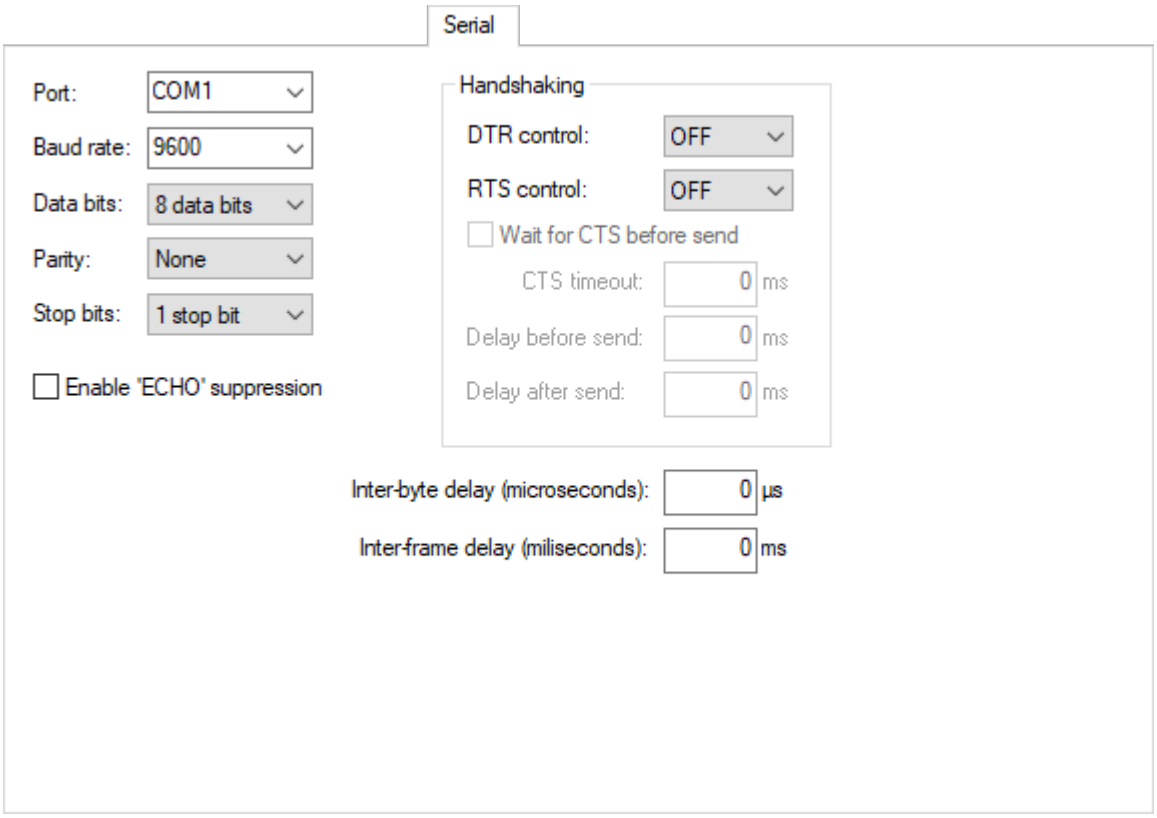
| OPTION | DESCRIPTION |
|---|---|
| **Mode** | Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection |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

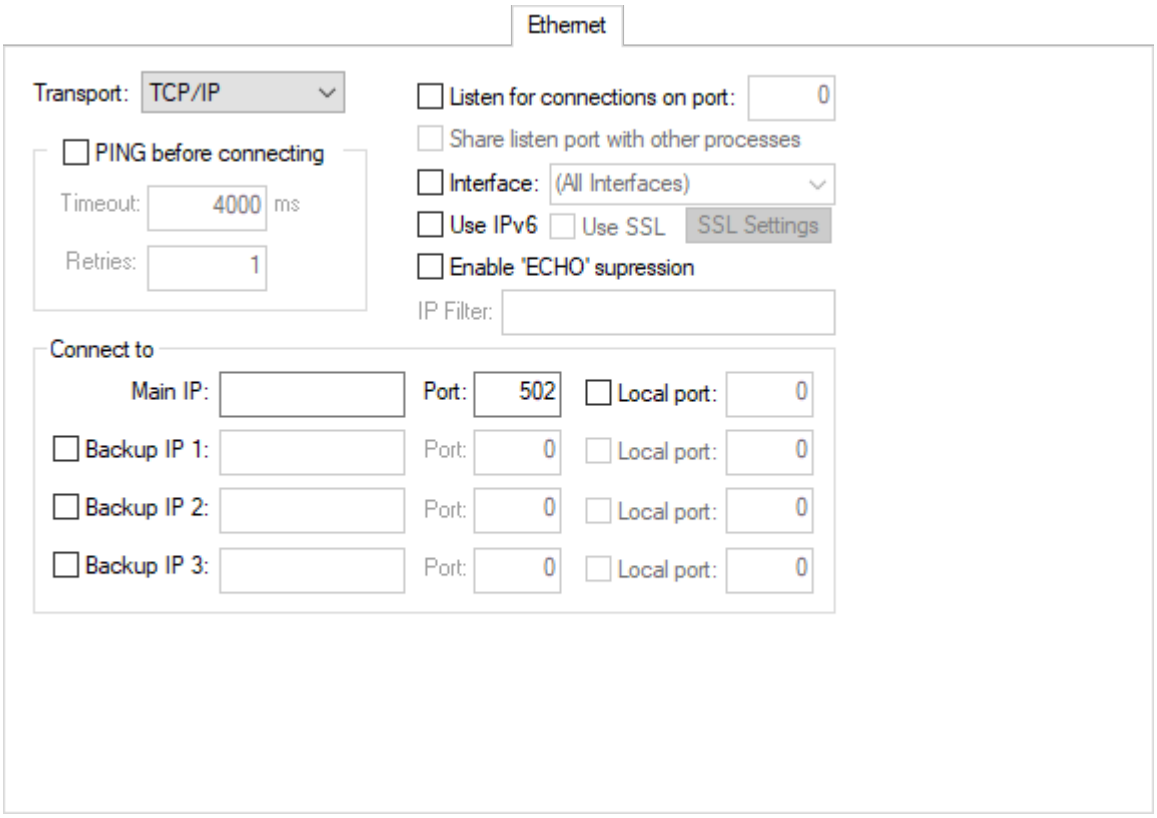
The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process, that is, it controls when data can be sent or received via serial line. Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works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

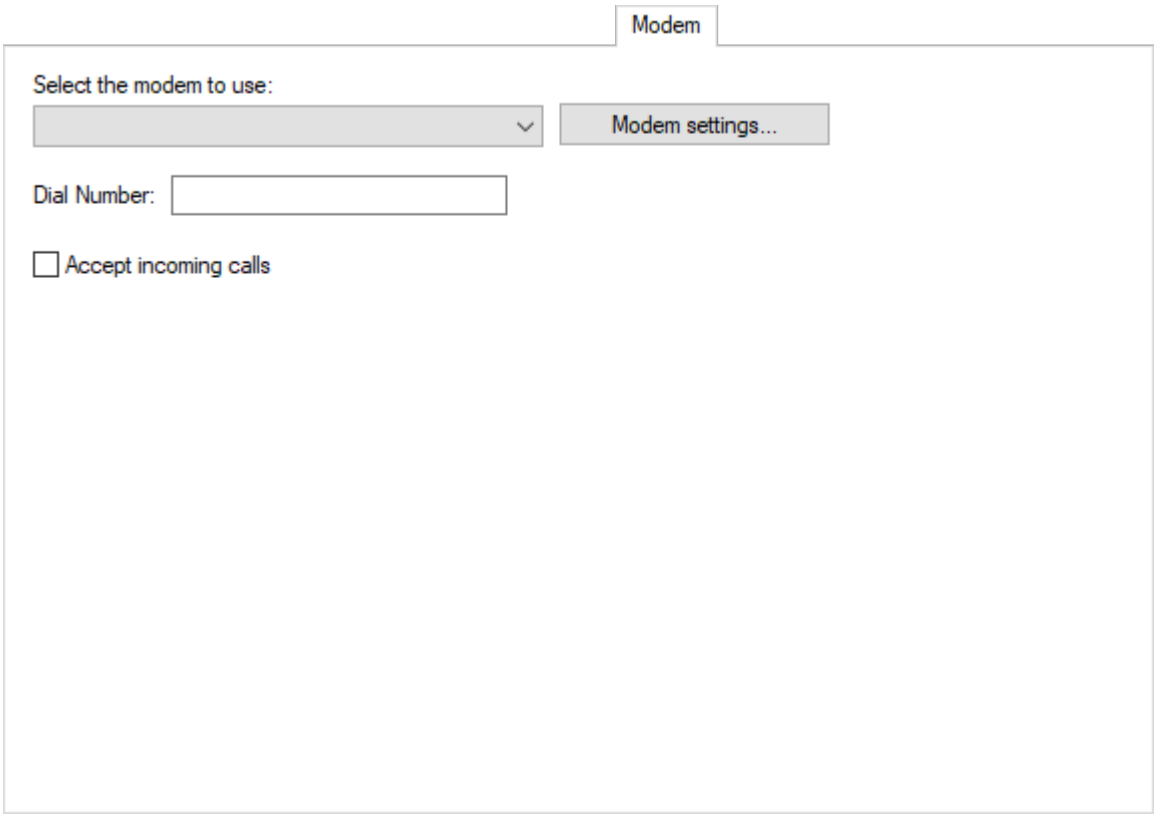
| OPTION | DESCRIPTION |
|---|---|
| | of determining a successful connection before trying to open a socket with a device. The time-out of a connection with a socket can be very high. The available options are:<br>• **Timeout**: Specify the number of milliseconds to wait for a reply from a **ping**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

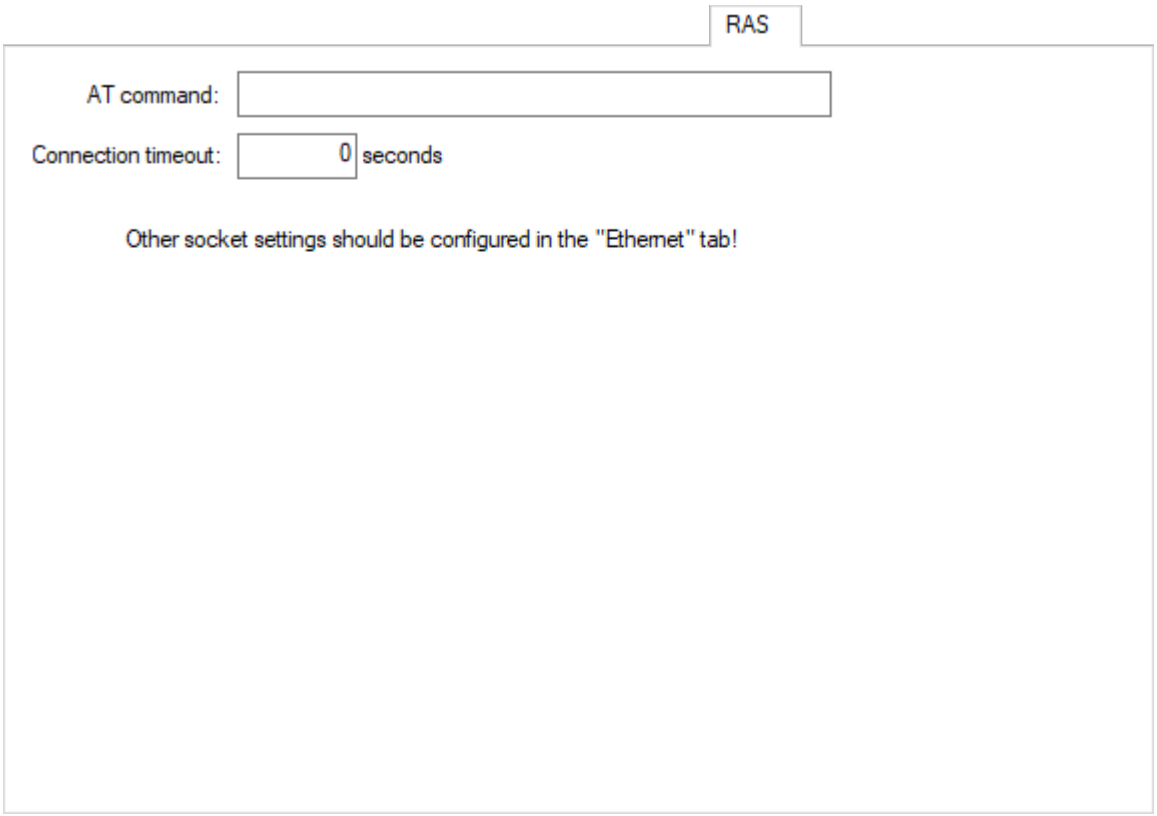
A **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

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]

A 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]

9 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]

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

## IO.Ethernet.IPFilter

A List with a comma-separated IPv4 or IPv6 addresses, which defines from which addresses 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

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's Revision History

| VERSION | DATE | AUTHOR | COMMENTS |
|---|---|---|---|
| **4.0.1** | 02/13/2026 | M. Ludwig | • Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 38173*). |
| **3.3.1** | 02/01/2012 | C. Mello | • Implemented support for reading and writing data in text format (*Case 12558*).<br>• Fixed a reading error with **X Contacts** (*Case 12697*).<br>• Implemented support for addressing devices in **Broadcast** mode using the *N1* parameter equal to 255. |
| **3.2.1** | 07/07/2010 | M. Ludwig | • Implemented compatibility with Windows CE (*Case 10889*). |

| VERSION | DATE | AUTHOR | COMMENTS |
|---|---|---|---|
| **3.1.1** | 10/28/2009 | M. Zani | • Fixed a problem when reading a single contact in a block with 16 elements (*Case 10594*). |
| **3.0.1** | 03/16/2009 | M. Zani | • Commands rewritten according to the protocol to allow blocks with more than 24 elements on a new range of values of the *N2* parameter, between 1000 and 1205.<br>• Added new commands.<br>• Implemented support for a floating point data type (*Case 9585*).<br>• Implemented support for a **DWord** data type (*Case 9474*).<br>• Implemented support for a **Long** data type (*Case 10073*). |
| **2.1.1** | 11/05/2007 | A. Quites | • Enabled block writing (*Case 8929*). |
| **2.0.1** | 12/07/2006 | A. Quites | • Implemented support for TCP/IP communication and Driver ported to **IOKit** library (*Case 3709*).<br>• Implemented support for TCP/IP communication using an **FP2 ET-LAN** module (Case 5369).<br>• Changed the format of addresses with a **Bit** data type to a hexadecimal format (*Case 3404*).<br>• Implemented reading data blocks in batches (*Case 6592*).<br>• Improvements in the process of checking consistency of received data, allowing an early detection of errors. |
| **1.0.1** | 01/15/2004 | R. Haetinger | • All versions previous to 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)