Moeller Easy Driver

File Name	Easy.dll
Manufacturer	Moeller Eaton Industries
Devices	Moeller Easy devices
Protocol	Proprietary
Version	3.0.1
Last Update	03/17/2026
Platform	Win32
Dependencies	IOKit version 1.15 or later
Superblock Readings	No
Level	0

Introduction

This Driver allows communication between **Elipse Software** systems and Easy devices by Moeller Eaton Industries, using a proprietary communication protocol. The physical connection uses a normal RS-232 cable, provided by Moeller Eaton Industries.

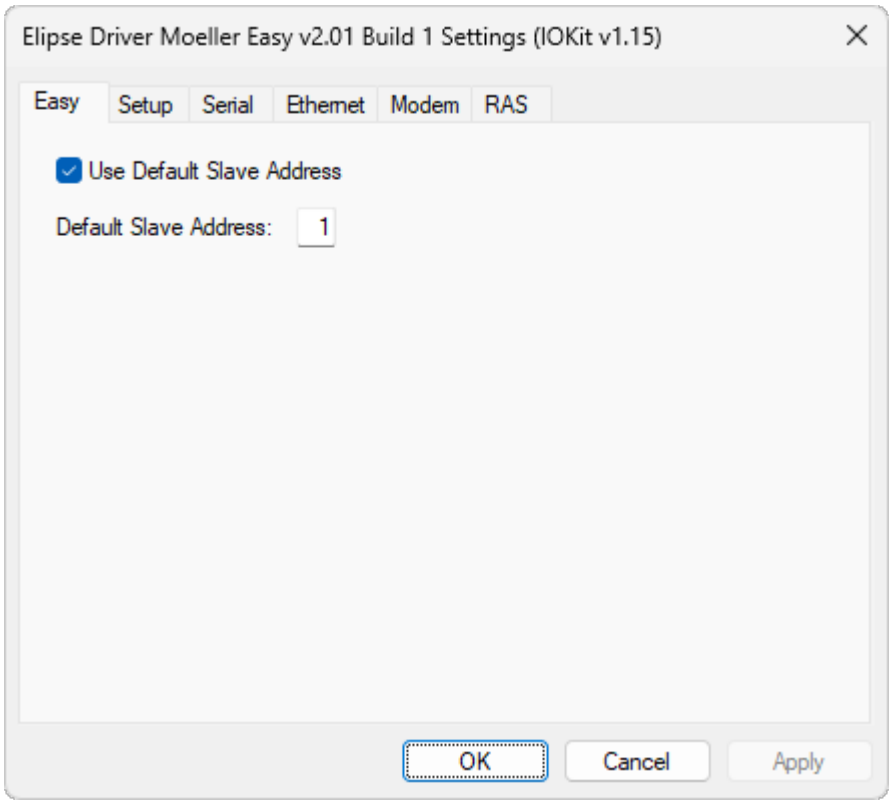
Driver Configuration

This section contains information about the configuring of **[P]** parameters of this Driver.

[P] 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)

All configurations are performed on the **Easy** tab on the properties window of this Driver, shown on the next figure.



Easy tab

The **Use Default Slave Address** option allows configuring a default value to use as the address of a **Slave** Easy device. This option is selected by default and the default value of the **Default Slave Address** option is 1 (one). If the **Use Default Slave Address** option is selected, any value used in the *N1* or *B1* parameters of this Driver's Tags are disregarded.

NOTE

The maximum number of Elements of a Block Tag is 8 (eight, the maximum number supported by the protocol). If a Block Tag is created with more than this limit, the Elements exceeding this limit are ignored.

Tag Reference

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

N and B Parameters for Tag Addressing

N1 or B1	Address of a **Slave** Easy device, between 1 (one) and 15
N2 or B2	Type of object for reading or writing or the number of a command. For more information, please check tables **Available commands** and **Reading or writing**
N3 or B3	Address, used only for reading or writing and not for commands
N4 or B4	Not used, leave it in 0 (zero)

Commands

Available commands

N2 OR B2	N3 OR B3	COMMAND
C0h	-	Stop
C1h	-	Run
C2h	-	Enable Write Program (Disable Editor)
C3h	-	Disable Write Program (Check EEPROM)
C4h	-	Reset System Test (Watchdog)
C5h	-	Toggle Remanenz on or off
C6h	-	Toggle Anlauf Stop or Run
C7h	-	Toggle LED
C8h	-	Turn P-Keys on
C9h	-	Turn P-Keys off
Cah	-	Turn *I-Entprellung* on
CBh	-	Turn *I-Entprellung* off
CCh	-	Turn Daylight Savings on
CDh	-	Turn Daylight Savings off
Ceh	-	Turn Remote Mode on
CFh	-	Clear Display in Remote Mode
D0h	-	Turn Remote Mode off
D1h	-	*Hole vierstelligen Wert im Remotemodus*
D2h	-	Enable *Auftragsbuch Sonder-Objekt*
D3h	-	Disable *Auftragsbuch Sonder-Objekt*

Reading or Writing

Reading or writing

N2 OR B2	N3 OR B3	TYPE OF OBJECT
00h	Address	Program
01h	Address	Dynamic Data, Identification, and Status
02h	Address	Internal clock
03h	Address	Display, only in Remote Mode
04h	Address	Special memory for Debugging and Monitoring
05h	Address	*Auftragsbuch für Sonderspeicher*

NOTE

For a list of objects and their respective addresses, please check topic **Sucom Easy Tips**.

Examples

If users want to access digital outputs of an Easy device, the parameters of a PLC Tag must be the following:

- **N1**: 1 (one, **Slave** with identifier one)
- **N2**: 01h (dynamic data, identification, and status)
- **N3**: 16h (for more information, please check topic **Sucom Easy Tips**)
- **N4**: 0 (zero, not used)

If users want to access *Merkers* from 0 (zero) to 7 (seven), the parameters of a PLC Tag must be the following:

- **N1**: 1 (one, **Slave** with identifier one)
- **N2**: 01h (dynamic data, identification, and status)
- **N3**: 14h (for more information, please check topic **Sucom Easy Tips**)
- **N4**: 0 (zero, not used)

If users want to read and set the internal clock of an Easy device, the parameters of a Block Tag must be the following:

- **B1**: 1 (one, **Slave** with identifier one)
- **B2**: 02h (internal clock)
- **B3**: 00h (for more information, please check topic **Sucom Easy Tips**)
- **B4**: 0 (zero, not used)
- **Size**: 3 (three, for more information, please check topic **Sucom Easy Tips**)

According to topic **Sucom Easy Tips**, the Elements of a Block Tag for the internal clock are the following:

- Weekday, between 0 (zero) and 6 (six)
- Hour, between 0 (zero) and 23
- Minute, between 0 (zero) and 59

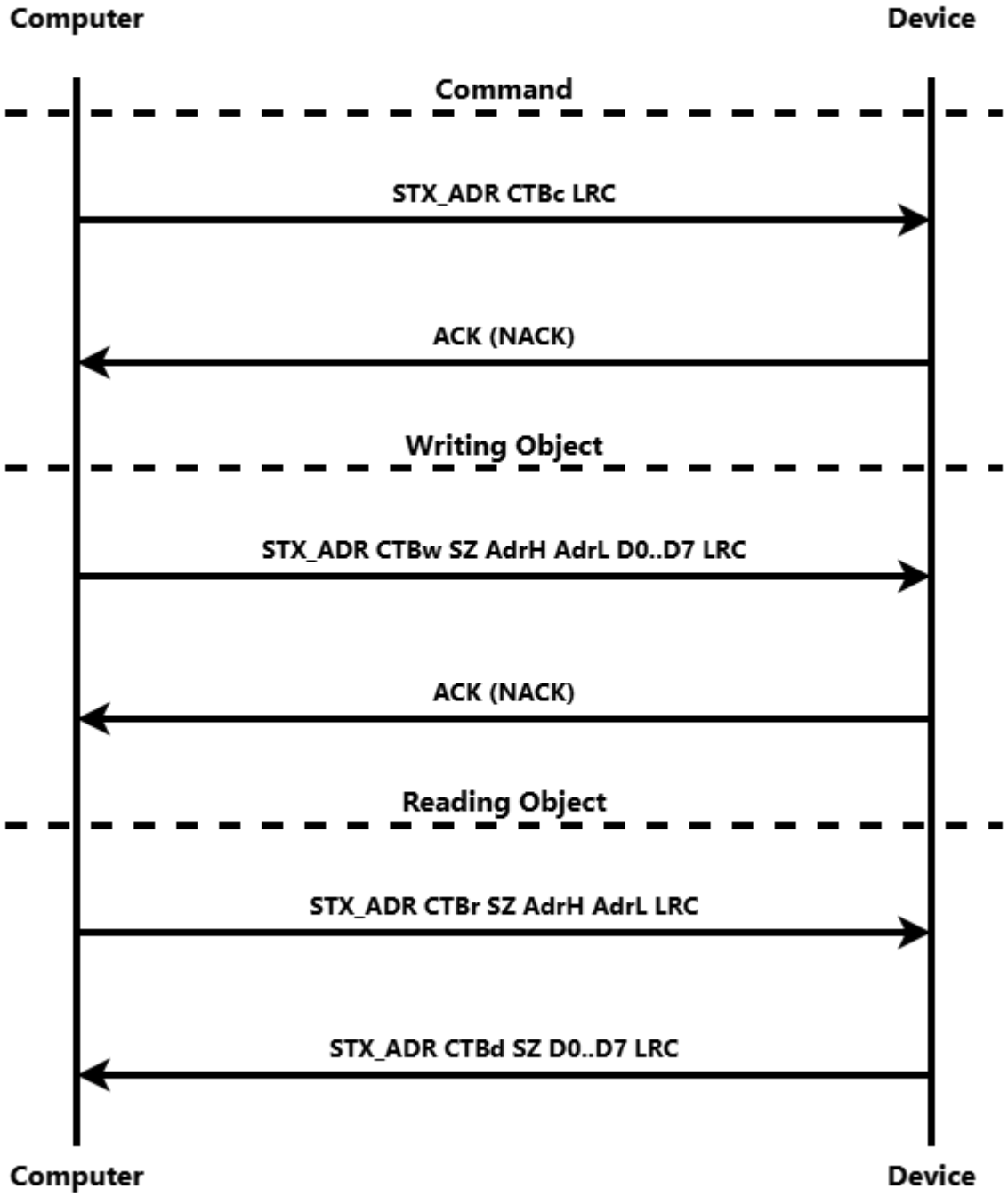
Users can read these Elements separately but, to set the internal clock, a Block Tag must be written at once.

Sucom Easy Tips

SUCOM-E

Transferring data to a computer

The protocol described here allows communication between the serial port RS-232 of a computer and the interface of an Easy controller at a 4800 bauds, 8 (eight) data bits, and 1 (one) stop bit, without parity check.



Protocol

```
STX_ADR = 1100aaaaB Control byte PC -> Easy Command
          ++++----- Address Coding (16)
                          00000 : PC
                          00001 : EASY
```

The **STX** byte always starts communication and it contains the address of an Easy device. Therefore, this protocol supports multi-drop operations, although this feature is not yet supported by the serial port RS-232.

The Easy programming interface provides the basic services described next, encoded in a control byte, which is the first byte to transfer.

Commands in which the control code starts with the value C0h (up to 32 commands are allowed)

```
CTBc = 11000000B Control byte PC -> Easy Command
       |||+++++--Command Coding (32)
       |||       00000: Halt
       |||       00001: Start
       |||       00010: Enable WRITE Program (-> Disable Editor)
       |||       00011: Disable Write Program (Check+->EEPROM)
       |||       00100: Reset System Test (Watchdog)
       |||       00101: Toggle retention ON/OFF
       |||       00110: Toggle start STOP/RUN
       |||       00111: Toggle LED at 4xx (not implemented at 6xx)
       |||       01000: P-Buttons ON
       |||       01001: P-Buttons OFF
       |||       01010: I-Debouncing ON
       |||       01011: I-Debouncing OFF
       |||       01100: Daylight Saving Time ON
       |||       01101: Daylight Saving Time OFF
       |||       01110: Turn on Remote Mode
       |||       01111: Erase display in Remote Mode
       |||       10000: Turn off Remote Mode
       |||       10001: Fetch four-digit value in Remote Mode
       |||       10010: Enable order book special object
       |||       10011: Disable order book special object
+++------- Command
```

A computer must transfer an **EASY HALT** command using the next sequence.

```
STX_ADR (C1h), CTBc (C0h), LRC (C0h)
```

The **LRC** byte is equal to the sum of data between **STX_ADR** and **LRC** bytes.

Memory object containing a reading, a control code addressed starting at 80h, and a writing, a control code addressed starting at A0h (up to 32 objects can be addressed)

```
CTBr = 10000000B Control byte PC -> Easy "Read object from ADDRESS"
CTBw = 10100000B Control byte PC -> Easy "Write object from ADDRESS"
       |||+++++-- Object Coding (32 objects)
       |||        00000: Program
       |||        00001: Images (dynamic data + ID, status)
       |||        00010: Set clock
       |||        00011: Display (only in Remote Mode)
       |||        00100: Special memory for debugging or visualization
       |||        00101: Order book for special storage
+++------  0 = Read Object
           1 = Write Object
```

The other structure of that message consists, primarily, of 3 (three) control bytes. In **SZ**, the number of data bytes is indicated, and it can reach 8 (eight). A pointer to a 16-bit address is added to the beginning of an object. Each object can can be up to 64 Kb in size.

```
SZ   = number of data bytes 1-8 (8 bytes)
AdrH = starting address relative to the beginning of the object (High)
AdrL = starting address relative to the beginning of the object (Low)
```

This is followed by a data message (letter to an Easy device) with up to 8 (eight) data bytes and a check sum (**LRC**), which contains the sum starting at **AdrL**. An Easy device responds to commands with an **ACK** (*Acknowledge*), a **NACK** (*NoAcklowledge*), or a **REJ** (*Rejection*) byte.

```
REJ  01010000b ; REJECT+ Adr.PC
NACK 00110000b ; NACK+ Adr.PC
ACK  01100000b ; ACK + Adr.PC
```

If a data package is requested, the protocol answers in the next format.

```
CTBd 00010100b ; 14h: Easy -> PC "Data Received";
D0.D7 with up to 8 data bytes; LRC
```

Composition of objects and formats

The header of a program object contains, among other data, set point values of system blocks.

```
Offset        Description
=========================
Program Header
0000h          2 Bytes Program length (00 = Program deleted) LOW, HIGH
0002h          1 Byte Parameter length (094h = 148 bytes)
0003h          1 Byte Program password
0004h          1 Byte minimum EASY hardware required by the program
              00000000
              |||||||+-- Analog:   1=TRUE 0=FALSE
              ||||||+--- Hour:     1=TRUE 0=FALSE
              |||||+---- I16_Q8:   1=TRUE 0=FALSE
              ||||+----- S_R_Data: 1=TRUE 0=FALSE
              |||+------ P-buttons used in the program
              ||+------- Retention Flag
              |+-------- 0=P-Buttons: 0= disabled 1=enabled
              +--------- 0=I-Debouncing: 0=YES 1=NO
0005h          8 bytes Program name
000Dh          1 byte Check sum header(the first 16 bytes 0-f ADDC )
000Eh          1 byte Program check sum
000Fh          1 byte Parameter field check sum
Parameter field (set point)
0010h          1 byte Control byte of the timer module T0
              00000000
              |||||+++-- 000 = X Raising delay (on delay)
              |||||      001 = ° Release delay (off delay)
              |||||      010 = ?X Random - Raising delay
              |||||      011 = ?° Random - Release delay
              |||||      100 = _U_ Clock Generator
              |||||
              |||++----- Time base:
              |||        00 = 10 ms
              |||        01 = seconds
              |||        10 = minutes
              |||        11 = not implemented
              |||
              ||+------- not implemented
              |+-------- 0= - removes from parameter menu
              |          1= + adds to parameter menu
              +--------- 0= stopped
                         1= in progress
0011h          2 bytes T0: Set point in 10 ms:    S:  (S*100+ms)LOW,HIGH
                       T0: Set point in seconds:  M:S (M*60+S)  LOW,HIGH
                       T0: Set point in minutes:  H:M (H*60+M)  LOW,HIGH
0013h          Parameter T2
0016h          Parameter T3
0019h          Parameter T4
001Ch          Parameter T5
001Fh          Parameter T6
0022h          Parameter T7
0025h          Parameter T8
0028h          1 byte Control byte of counter C0
              00000000
              |+-------- 0= - removes from parameter menu
              |          1= + adds to parameter menu
              +--------- 0= stopped
                         1= in progress
0029h          2 Bytes C0: Set point: LOW,HIGH (0-9999)
002Bh          Parameter C2
002Eh          Parameter C3
0031h          Parameter C4
0034h          Parameter C5
0037h          Parameter C6
003Ah          Parameter C7
003Dh          Parameter C8
0040h          1 byte W1 Control Byte
              00000000
              +--------- 0= stopped
                         1= in progress
0041h          W1 1 byte Control byte of Channel 1, Weekday Start-End
              00000000
              |||||+++-- Starting Date
```

```
              |||||      000 = Channel: not edited
              |||||      001 = Mo
              |||||      010 = Tu
              |||||      011 = We
              |||||      100 = Th
              |||||      101 = Fr
              |||||      110 = Sa
              |||||      111 = Su
              ||+++----- End Date
              ||         000 = Disable
              ||         001 = Mo
              ||         010 = Tu
              ||         011 = We
              ||         100 = Th
              ||         101 = Fr
              ||         110 = Sa
              ||         111 = Su
              ||
              |+-------- 1= Ton>Toff
              |          0= Ton<=Toff or Ton or Toff
              |             "Off" = (Bit.15=1))
              +--------- 0= - remove from parameter menu
                         1= + add to parameter menu
0042h          W1 2 bytes Channel 1, Hour/Minute ON (Low,High) Format: Integer "HH*60+MM"
                "Off" is Bit.15=1 and 10-Date =3
0044h          W1 2 bytes Channel 1, Hour/Minute OFF (Low,High.7=1:Off) Format: Integer "HH*60+MM"
                "Off" is Bit.15=1 and 10-Date =3
0046h          W1 1 byte control byte for Channel 1, Weekday Start-End
0047h          W1 2 byte Channel 2, Hour/Minute ON
0049h          W1 2 byte Channel 2, Hour/Minute OFF
004Bh          W1 1 byte control byte for Channel 2, Weekday Start-End
004Ch          W1 2 byte Channel 3, Hour/Minute ON
004Eh          W1 2 byte Channel 3, Hour/Minute OFF
0050h          W1 1 byte control byte for Channel 3, Weekday Start-End
0051h          W1 2 byte Channel 4, Hour/Minute ON
0053h          W1 2 byte Channel 4, Hour/Minute OFF
0055h          Parameter W2
006Ah          Parameter W3
007Fh          Parameter W4
0094h          1 byte Control byte of analog comparator A1
              00000000
              |||||||+-- 0 = >=
              |||||||    1 = <=
              |||||++--- Comparator Mode
              |||||      00 = IA1 : IA2
              |||||      01 = IA1 : Const
              |||||      10 = IA2 : Const
              |||||      11 =
              |+-------- 0= - remove from parameter menu
              |          1= + add to parameter menu
              +--------- 0= stopped
                         1= in progress
0095h          1 byte A1: Constant: (Format: 0-99)
0096h          Parameter A2
0098h          Parameter A3
009Ah          Parameter A4
009Ch          Parameter A5
009Eh          Parameter A6
00A0h          Parameter A7
00A2h          Parameter A8
00A4h          4 bytes Compensation/Offset for pagination limit (16 Bytes) for serial EEProms
   User program (diagram start)
00A8h          8 bytes per row * 121 rows (968 bytes)
0470h          and another 2 spare rows
   MMI Text Modules
0480h          8 texts with 80 bytes of control codes and text (640 bytes)
06FFh          End of the program object
Writing to the following addresses is not blocked. So, crashes may occur during this operation
```

An **image** object contains the current data for inputs, outputs, and flags, among others, and also real-time values from system modules.

```
Offset        Description
=========================
   Current Configuration (determined by the self-test)
   Resources provided by the device
0000h          1 byte Device Configuration / type identifier
              00000000
              |||||||+-- Analog:     1=TRUE      0=FALSE
              ||||||+--- Clock:      1=TRUE      0=FALSE
              |||||+---- I16_Q8:     1=TRUE      0=FALSE
              ||||+----- S_R_Data:   1=TRUE      0=FALSE
              |||+------ Start:      1=TRUE      0=FALSE
              ||+------- 1=1. Time query language
              |+-------- 1=DST ID
              +--------- AC/DC:      1=AC        0=DC
              Type         Code   Type          Code
              ====================================
              412-DC-R     01h    412-AC-R      80h
              412-DC-RC    03h    412-AC-RC     82h
              412-DC-TCX   03h    412-AC-RCX    82h
              620-DC-TC    0Fh    618-AC-RC     8Eh
   EASY Device Status
0001h          1 byte status information
              00000000
              |||||||+-- Status:     1=RUNNING    0=HALTED
              ||||||+--- Password:   1=Password active
              |||||+---- Program:    1=Schematic available
              ||||+----- Test:       1=Hardware test active
              |||+------ Card:       1=Memory Card plugged
              ||+------- I2C:        1=I2C busy, EEPROM access
              |+-------- Save Param  1=Save parameters (EEPROM)
              +--------- Save Prog   1=Write program (EEPROM) 620-DC-TCX  0Fh    618-AC-RCX    8Eh
   EASY Errors/Configuration
0002h          1 byte Error Flags and part 2 of configuration
              00000000
              |||||||+-- EEPROM:   EEPROM failure
              ||||||+--- EEAck:    EEPROM is not responding
              |||||+---- Clock:    Clock failure
              ||||+----- LCD:      Display failure or display does not exist
              |||+------ ACLow:    AC level below the necessary
              ||+------- Network:  Disconnected or not available
              |+-------- X-Type:   1=Type without Display (s. o.)
              +--------- InitHW:   1=Hardware initialization
   Reference Configuration (required by the program)
   Resources required by the device program
0003h          1 byte Configuration Flags
              00000000
              |||||||+-- Analog:       1=Analog devices in use
              ||||||+--- Clock:        1=Clock in use
              |||||+---- I16Q8:        1=I>8 program in use
              ||||+----- SR-Operand:   1=Send/Receive in use
              |||+------ P-Buttons:    1=P-Buttons in use
              ||+------- Retention:    1=Retention enabled
              |+-------- P-Buttons 1:  1=P-Buttons enabled
              +--------- Debouncing:   1=Debouncing disabled
   Version / Language
0004h     1 byte OS version (10h = V1.0)
0005h     1 byte Display language
         0: English, 1: German, 2: French, 3: Spanish, 4: Italian,
         5: Swedish, 6: Dutch, 7: Polish, 8: Portuguese, 9: Turkish
Image Data
0006h          3 bytes Clock reference
              WD, Weekdays 0-6
              HH, Hour 0-23
              MM, Minute 0-59
0009h          2 bytes Reference of analog inputs I7,I8 (Format: 0-99)
000Bh          1 byte Analog Constant 0-k0 / N * k0 = FF (must be "0)
000Ch          2 bytes Image of inputs from I0 to I16
000Eh          1 byte free
000Fh          1 byte P-Buttons Image
              00000000
              ||||||||
```

```
             |||||||+----------- Tleft
             ||||||+------------ Tup
             |||||+------------- Tdown
             ||||+-------------- Tright
             |||+--------------- Tesc
             ||+---------------- Tok
             |+----------------- Tdel
             +------------------ Tpen
0010h         1 byte Output image of the Timer
             OutputTimer: 0-7 (Bit 0 = T1 etc.)
0011h         1 byte Output Image of the Counter
             OutputCounter: 0-7
0012h         1 byte Output Image of the Clock Module W
             OutputClockWatch: 0-3
0013h         1 byte Output Image of the Analog Block A
             OutputIAnalogComp: 0-7
   More Image Data
0014h        Image_M0: Flag: 0-7
0015h        Image_M8: Flag: 8-F
0016h        Image_Q0: Digital Output: 0-7
0017h        Image_S0: Send Output Communication: 0-7
0018h        Image_D0: Text flag: 0-7
0019h        Image_R0: Receive Communication: 0-7
001Ah        Image_R8: Receive Communication: 8-F
   Image Data for Timer Modules
001Bh        T1: Status flags
            00000000   Control Byte
            ||||||||
            ||||||++-- 000 =  X Raising delay.(ON delay)
            ||||||     001 =  °  Release delay.(OFF delay)
            ||||||     010 = _|- Random raising delay
            ||||||     011 = _U_ Clock generator
            ||||||
            |||||+---- 0= MS
            |||||      1= SECONDS
            ||||+----- Reset-SET
            |||+------ "SET"-Status = SET-ALT
            ||+------- "RES"-Status
            |+-------- "Tx"-Status
            +--------- 0= stopped
                       1= increment on INTERRUPT-TIMER
001Ch         2 bytes T1: Timer's instant value
001Eh         1 byte  T1: Random value
001Fh        Timer 2 w.o.
0023h        Timer 3
0027h        Timer 4
002Bh        Timer 5
002Fh        Timer 6
0033h        Timer 7
0037h        Timer 8
   Image data for counter modules
003Bh       1 byte C1: Status Flag
            2 byte C1: Counter value
003Eh       Counter 2 w.o.
0041h       Counter 3
0044h       Counter 4
0047h       Counter 5
004Ah       Counter 6
004Dh       Counter 7
0050h       Counter 8
   Internal Data
0053h      1 byte FB Watch: Compares Minutes
0054h      1 byte NetIdentCode: Identification of the connected extension
          0 Error, 1 AC Expansion, 2 DC Expansion, 3 Easy200Easy, 4 Network participants
0054h      End of the Image Object
          Writing above this limit may lead to an undefined behavior on the CPU
The Uhr (clock, in German) object contains the real-time clock
Offset         Description
==========================
0000h         3 bytes Clock Reference
              WD, Weekdays 0-6
```

```
            HH, Hours 0-23
            MM, Minutes 0-59
Writing to this object performs a time definition
With the following sequence of commands, users can set the current time to Monday, 12:00:
STX_ADR (C1h), CTBw Clock-Object (A2h), Number (03h), OffsetH (0), OffsetL (0),
WK (0=Monday), HH (12d), MM (0), LRC (B1h)
An Easy device sends an ACK (B0h) when the command is executed
```

The **special memory** object is used to optimize data transfer for viewing.

```
Offset        Description
==========================
Byte array, where each byte represents a data area of a corresponding input in the order book
0000h      Size of the array
0001h      Data related to the first array in the order book
0002h      Data for the second array, etc.
Memory is dynamically managed
```

The **order book special memory** object controls the inputs that a computer writes to the **special memory** object.

```
Offset        Description
==========================
Handled using a word array
0000h      First element of special memory viewing
0002h      Second... etc.
Format:
aaaa aaaa-aaao oooo
|||| |||| |||| ||||
|||| |||| |||+-++++- Object Code
++++-++++-+++------- 12 bit Offset address
Memory is dynamically managed. The 1Fh code indicates the object's end
Sending the e-SUCOM command "Enables an order book special object (10010)" overwrites this code
Codes (bytes) of viewing objects in the order book:
00000: Program
00001: Images (dynamic data + ID, status)
00010: Set clock
00011: Display (only in Remote Mode)
00100: Special memory for debugging/viewing
00101: Order book for special storage
```

NOTE

Invalid formats and data content sent to an Easy device may result in crashes or unwanted behavior, such as output triggering. This may cause injuries or lead to death.

Documentation of I/O Interfaces

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

Configuration of a Driver

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

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

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

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

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

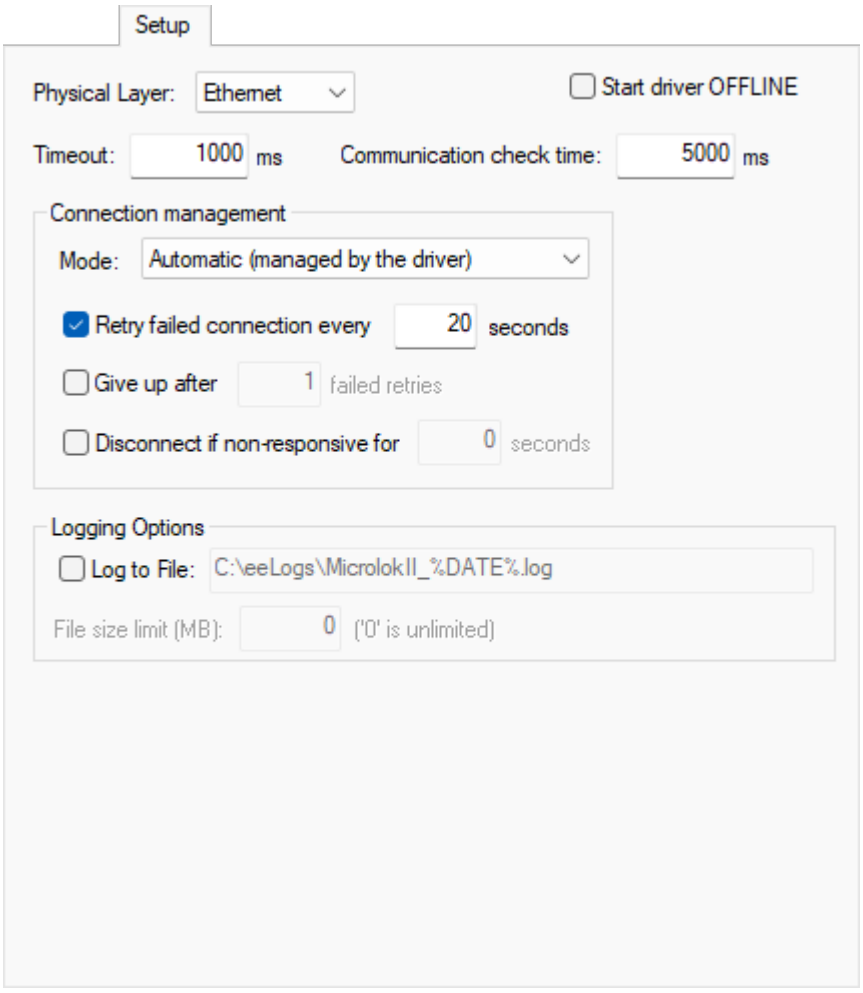
Configuration Dialog Box

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

Setup Tab

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

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



Setup tab

General options on the Setup tab

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

Options on the Connection management group

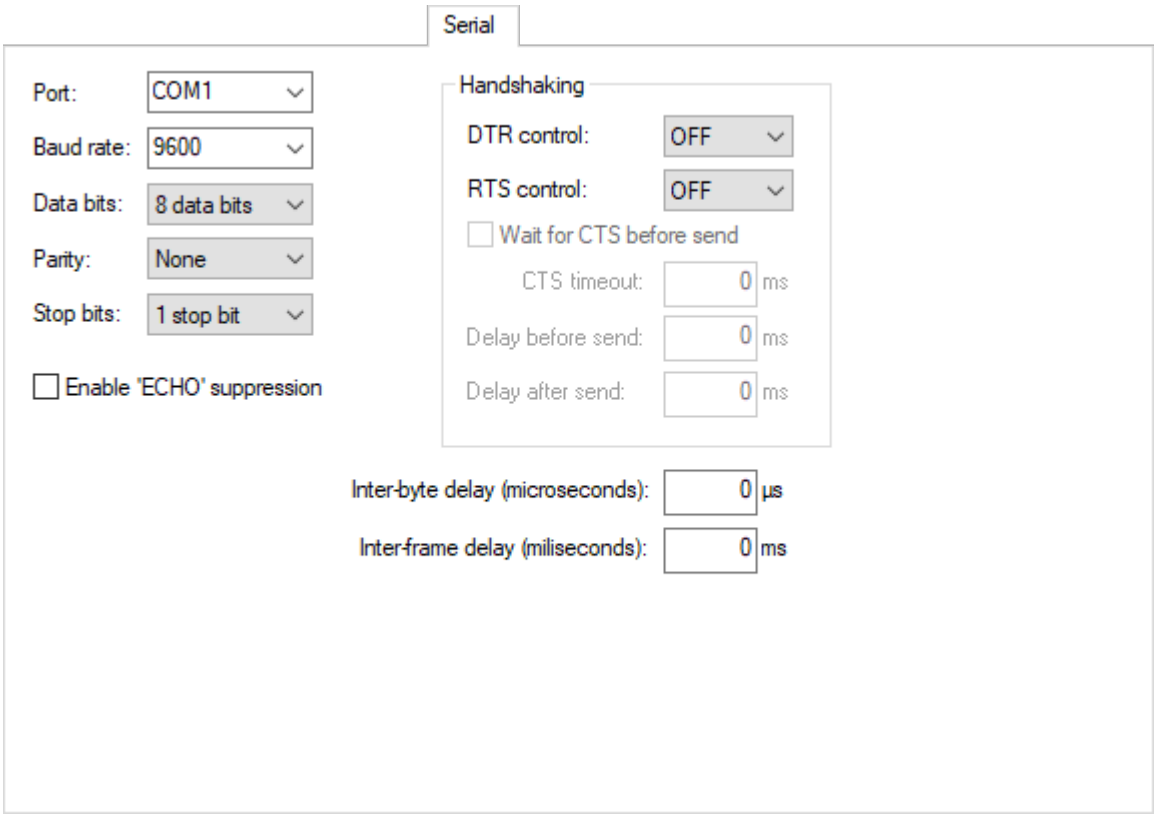
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection
Retry failed connection every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

Options on the Logging Options group

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

Serial Tab

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



Serial tab

General options on the Serial tab

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

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

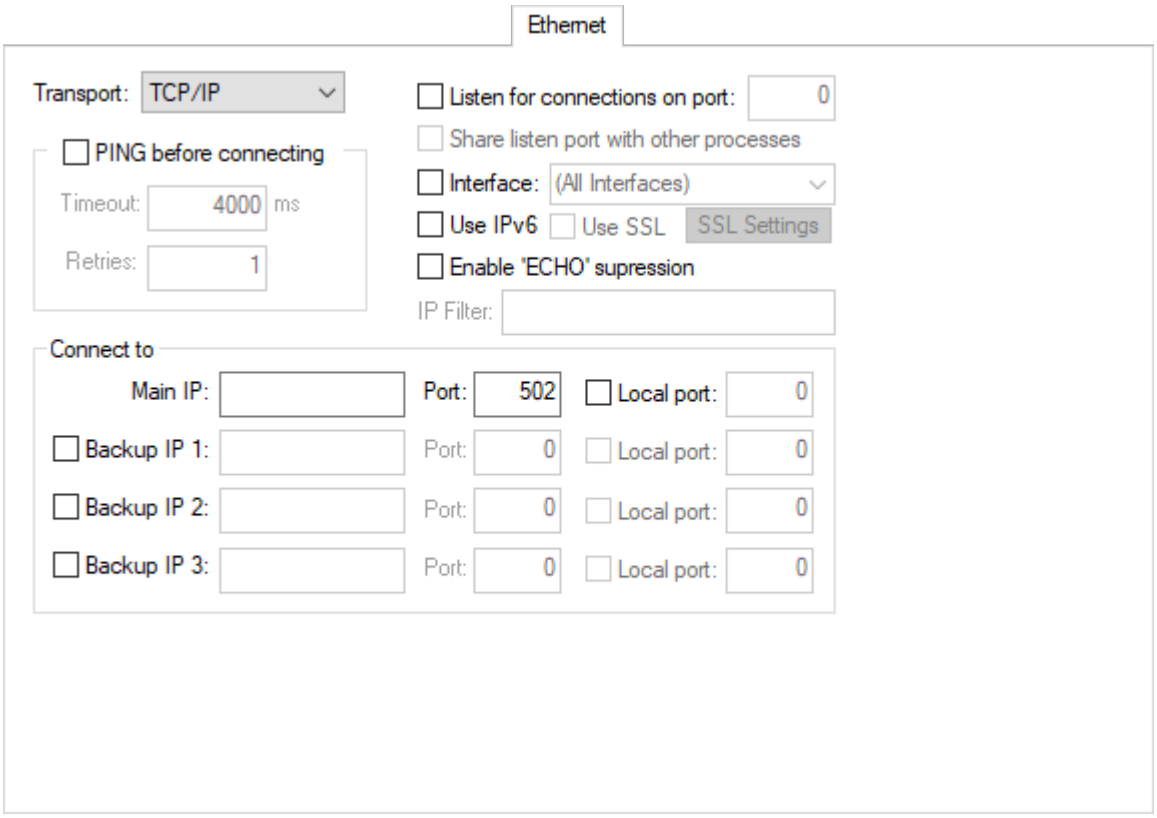
The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process, that is, it controls when data can be sent or received via serial line. Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works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

Available options on the Handshaking group

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

Ethernet Tab

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



Ethernet tab

Available options on the Ethernet tab

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

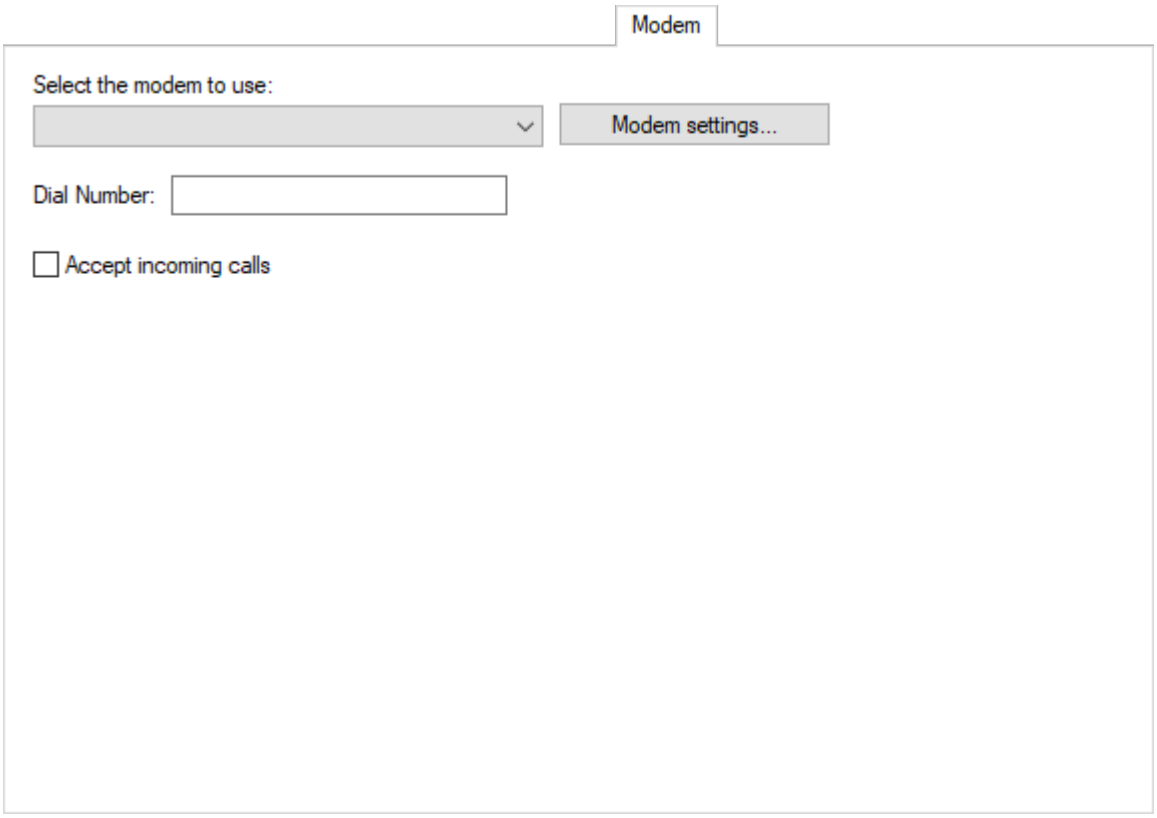
OPTION	DESCRIPTION
	of determining a successful connection before trying to open a socket with a device. The time-out of a connection with a socket can be very high. The available options are: • **Timeout**: Specify the number of milliseconds to wait for a reply from a **ping** command. Users must use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

Available options on the Connect to group

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

Modem Tab

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



Modem tab

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

Available options on the Modem tab

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

RAS Tab

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

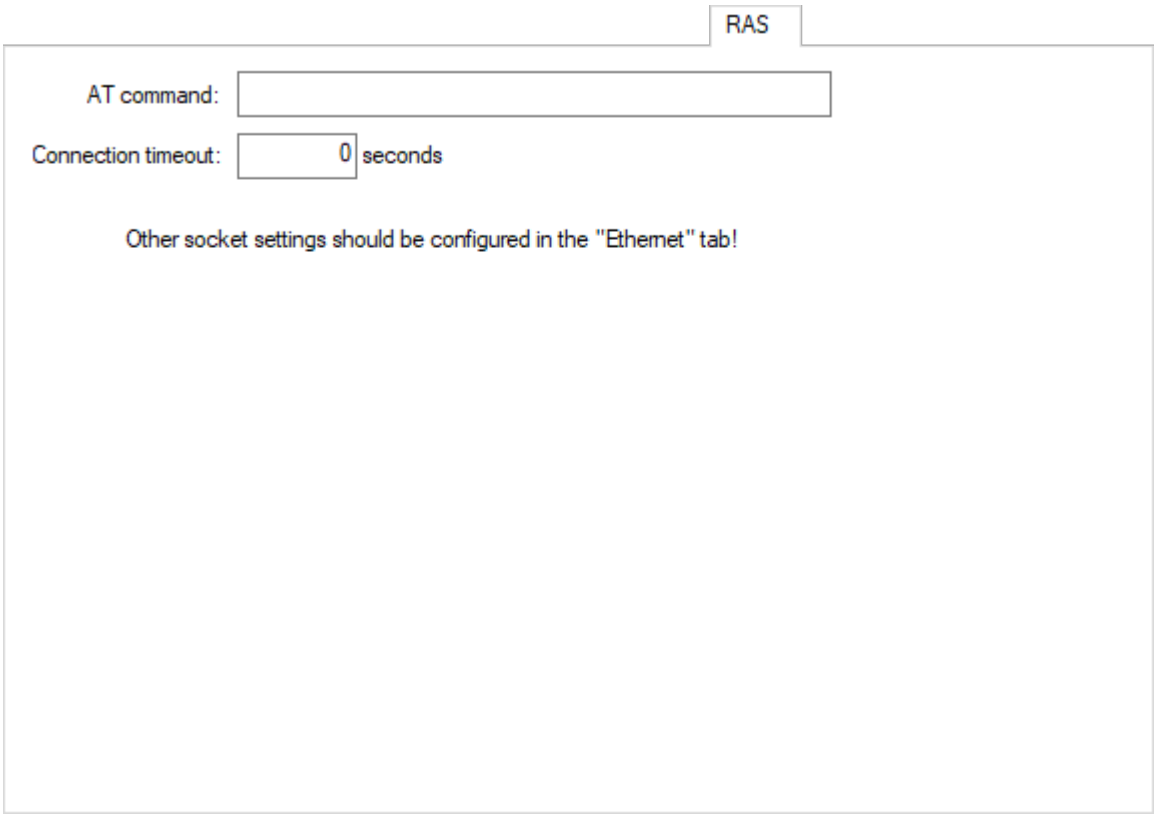
A **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

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

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

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

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



RAS tab

Available options on RAS tab

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

General Configurations

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

I/O Tags

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

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

IO.CommunicationStatus

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

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

IO.IOKitEvent

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

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

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

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

NOTE

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

IO.PhysicalLayerStatus

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

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

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

IO.SetConfigurationParameters

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

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

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

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

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

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

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

A variation of the previous example uses a bidimensional array.

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

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

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

IO.WorkOnline

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

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

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

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

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

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

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

IMPORTANT

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

Properties

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

IO.ConnectionMode

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

IO.GiveUpEnable

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

IO.GiveUpTries

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

IO.InactivityEnable

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

IO.InactivityPeriodSec

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

IO.RecoverEnable

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

IO.RecoverPeriodSec

Delay time between two connection attempts, in seconds.

NOTE

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

IO.StartOffline

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

NOTE

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

IO.TimeoutMs

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

IO.Type

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

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

Statistical Configuration

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

I/O Tags

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

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

IO.Stats.Partial.BytesRecv

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

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

IO.Stats.Partial.BytesSent

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

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

IO.Stats.Partial.TimeConnectedSeconds

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

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

IO.Stats.Partial.TimeDisconnectedSeconds

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

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

IO.Stats.Total.BytesRecv

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

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

IO.Stats.Total.BytesSent

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

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

IO.Stats.Total.ConnectionCount

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

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

IO.Stats.Total.TimeConnectedSeconds

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

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

IO.Stats.Total.TimeDisconnectedSeconds

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

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

Properties

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

Ethernet Interface Configuration

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

I/O Tags

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

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

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

IO.Ethernet.IPSelect

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

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

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

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

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

IO.Ethernet.IPSwitch

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

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

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

IO.Ethernet.SocketState

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

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

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

Properties

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

NOTE

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

IO.Ethernet.AcceptConnection

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

IO.Ethernet.BackupEnable[2,3]

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

IO.Ethernet.BackupIP[2,3]

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

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

IO.Ethernet.PingTries

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

IO.Ethernet.ShareListenPort

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

IO.Ethernet.SupressEcho

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

IO.Ethernet.Transport

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

IO.Ethernet.UseIPv6

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

Modem Interface Configuration

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

I/O Tags

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

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

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

IO.TAPI.ConnectionBaudRate

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

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

IO.TAPI.Dial

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

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

IO.TAPI.HangUp

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

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

NOTE

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

IO.TAPI.IsModemConnected

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

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

IO.TAPI.IsModemConnecting

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

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

IO.TAPI.ModemStatus

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

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

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

IO.TAPI.PhoneNumber

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

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

Properties

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

IO.TAPI.AcceptIncoming

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

IO.TAPI.ModemID

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

NOTE

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

IO.TAPI.PhoneNumber

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

RAS Interface Configuration

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

I/O Tags

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

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

Properties

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

NOTE

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

IO.RAS.ATCommand

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

IO.RAS.CommandTimeoutSec

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
3.0.1	03/17/2026	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 38767*).
2.1.1	08/16/2013	G. Taschetto	• Translated topic **Sucom Easy Tips** from German to English (*Case 2845*).
2.0.1	04/23/2009	M. Zani M. Ludwig	• Driver ported to **IOKit** library (*Case 6815*). • Driver ported to Windows CE (*Case 10331*).
1.1.1	03/29/2004	C. Mello	• Fixed a problem with unstable communication, keeping the RTS signal always in the highest level.
1.0.1	08/15/2001	R. Haetinger	• Initial version of this Driver.

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

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

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

www.elipse.com.br

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

Gartner 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)