Ictel CT850 Driver

File Name	CT850.dll
Manufacturer	Ictel Instrumentação Controles e Telemetria LTDA
Devices	Telemetry Center
Protocol	Modbus
Version	1.0.4
Last Update	01/26/2026
Platform	Win32
Dependencies	IOKit version 2.0 or later
Superblock Readings	Yes
Level	0

Introduction

This Driver implements the Modbus protocol, allowing an application developed by **Elipse Software** to communicate with Telemetry Center-type devices by Ictel Instrumentação Controles e Telemetria LTDA.

Preparing a Device

The goal of this Driver is to perform all procedures to execute a telecommand routine with the Ictel **CT850** Control Panels and **AUTTOM** (**Radio 1** and **Radio 2**) Control Panels, according to the proprietary schema of each Control Panel.

CT850 Control Panel

The remote stations of the **CT850** Control Panel can receive telecommands via external application using the next memory areas:

- **0x9800**: Fixed Modbus memory address to perform telecommand requests
- **0xA000**: Fixed Modbus memory address to retrieve the success or failure response from the telecommand requested by address **0x9800**

The **CT850** Control Panel can only execute one telecommand at a time.

AUTTOM (Radio 1 and Radio 2) Control Panel

The Remote Stations of the **AUTTOM** Control Panel can receive telecommands via external application using the next **Radio 1** memory areas (for **Radio 2** memory areas, increase the **Radio 1** memory areas by 100, with the exception of memory **303**, which remains fixed):

- **301**: Fixed Modbus memory address to indicate a Remote Station
- **302**: Fixed Modbus memory address to indicate a type of telecommand
- **303**: Fixed Modbus memory address to indicate a control number or type of Remote Station, depending on the telecommand used
- **300**: Fixed Modbus memory address to start the trigger command
- **320**: Fixed Modbus memory address to get a success or failure response
- **321**: Fixed Modbus memory address to get a return code (48 for success or other codes for errors)
- **304-311 and 324-331**: Fixed Modbus memory regions for storing control Setpoints
- **312-319 and 332-339**: Fixed Modbus memory regions for storing blocking Setpoints
- **340**: Fixed Modbus memory address to indicate the number of a peripheral
- **341**: Fixed Modbus memory address to indicate the initial internal memory of a peripheral
- **342**: Fixed Modbus memory address to indicate the amount of memory
- **343**: Fixed Modbus memory address to assign a reference value to a peripheral

Driver Configuration

This Driver does not use **[P]** configuration parameters. All **IOKit** library configurations, and the ones specific to this Driver, must be performed on this Driver's configuration dialog box or on the **Extra Settings** option in **Elipse SCADA**.

On this properties dialog box, the **CT850** tab contains specific settings for this Driver. The other tabs refer to communication settings of **Elipse Software**'s **IOKit** library.

For more information about the configurations of **IOKit** library, please check topic **Documentation of I/O Interfaces**.

Configuring Properties

This topic contains information about the properties available on the **CT850** tab, already including the value of the **Strings** of offline properties, which can be user-programmed when starting an application in **Offline** mode.

For **Elipse E3**, **Elipse Power**, or **Elipse Water** applications, the value of these settings can also be defined at run time. To do so, start this Driver in **Offline** mode, by selecting the **Start driver OFFLINE** option on the **Setup** tab of the properties dialog box.

The configuration options of this Driver are listed on the next table.

Configuration options for Ictel CT850 Driver

TAB	PARAMETER	OFFLINE STRING	DATA TYPE	DESCRIPTION
Setup	Physical Layer	IO.Type	Text	Configures the physical interface. The available options are **Serial**, **Ethernet**, **Modem**, or **RAS**
	Timeout	IO.TimeoutMs	Number	A time limit, in milliseconds, to receive data from a device's response. For example, a value of 1000 defines a limit of 1 (one) second
Ethernet	Transport	IO.Ethernet.Transport	Text	Configures the transport layer. The available options are **TCP/IP** or **UDP/IP**
	Main IP	IO.Ethernet.MainIP	Text	IP address of a device, in the format **[0-255].[0-255].[0-255].[0-255].[0-255]**
	Port	IO.Ethernet.MainPort	Number	TCP/IP port of a device. Possible values range from 0 (zero) to 65535
CT850	Telecommand Time-out (s)	CT850.TeleCommandTimeout	Number	Time limit, in seconds, for a Control Panel to respond with a return code referring to the processing of the requested telecommand

All offline properties must be configured via PLC Tags in **String** format using the *N1* parameter equal to -1 (minus one), the *N2* parameter equal to 0 (zero), the *N3* parameter equal to 0 (zero), and the *N4* parameter equal to 3 (three). For more details and examples, please check topic **Documentation of I/O Interfaces**.

Tag Reference

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

Telecommands

This section contains Tags for handling telecommands.

Commands

Use PLC Tags to manipulate specific telecommands from Control Panels, using the parameters indicated on the next table.

Parameters for Control Panels

Device	Parameters of a device, in the format **ID:XXXXX:Rn:Tn**
Item	Parameters of a telecommand. For more information, please check table Telecommand Tags
N1 or B1	Not used
N2 or B2	Not used
N3 or B3	Not used
N4 or B4	Not used

In the **Device** parameter of these Tags, inform the identification data of the Control Panels using the format **ID:XXXXX:Rn:Tn**, as described next:

- **ID**: Identifier of a Modbus slave
- **XXXXX**: Type of Control Panel, according to the next values:
  - **CT850**: Ictel CT850 Control Panel
  - **AUTTOM_R1**: Radio 1 Auttom Control Panel
  - **AUTTOM_R2**: Radio 2 Auttom Control Panel
- **Rn**: Number of a Remote Station
- **Tn**: Type of Remote Station

The next table contains telecommands that can be used to interact with Control Panels, by using the **Item** parameter.

Telecommand Tags

ITEM	DESCRIPTION	OPERATION
STATUS	Returns information about the current state of this Driver. Possible values are 0: Idle (number) or 1: Processing telecommand (number)	Read-only

ITEM	DESCRIPTION	OPERATION
Vn:COMMAND	Sends a telecommand to the controller number indicated by **Vn**. The writing values of this Tag used to select the type of control to send are **67**: Performs the triggering of the control indicated in **Vn**, **68**: Performs a shut down of the control indicated in **Vn**, **69**: Select a control for manual indicated in **Vn**, or **70**: Select a control for automatic indicated in **Vn**. Use a **Vn:RESPONSE** Tag to check whether the processing of the telecommand was completed successfully or failed	Write-only
Vn:RESPONSE	A read-only Tag used to monitor the result of the last telecommand sent to the controller number indicated by **Vn**. In case of success, this Tag returns a 32-bit integer, with the upper bits set to 0 (zero) and the lower bits containing the value written by the **Vn:COMMAND** Tag. In case of failure, this Tag returns a 32-bit integer, with the upper bits containing the Control Panel error and the lower bits containing the value written by the **Vn:COMMAND** Tag	Read-only

Usage Example

Writing a telecommand for the **CT850** Control Panel with identifier equal to 1 (one, CT850), Remote Station equal to 3 (three, R3), Type of Remote Station equal to 2 (two, T2), and to trigger control 5 (five, V5):

- **Tag.ParamDevice**: 1:CT850:R3:T2
- **Tag.ParamItem**: V5:COMMAND
- **Writing value of this Tag**: 67 (control triggering)

Return of this telecommand:

- **Tag.ParamDevice**: 1:CT850:R3:T2
- **Tag.ParamItem**: V5:RESPONSE
- **Reading value of this Tag in case of success**: 67, that is, a 32-bit integer 0x00000043, with the upper bits equal to 0 (zero) and the lower bits equal to 67
- **Reading value of this Tag in case of failure**: 3407939, that is, a 32-bit integer 0x00340043, with the upper bits equal to 52 and the lower bits equal to 67, in which the upper bits receive the error code 0x0034 from the Control Panel

Setpoints

Use PLC Tags to manipulate specific telecommands from Control Panels, using the parameters indicated on the next table.

Parameters for Control Panels

Device	Parameters of a device, in the format **ID:XXXXX:Rn:Tn**
Item	Parameters of a telecommand. For more information, please check tables **Telecommand Tags for Setpoints in CT850 Control Panels** and **Telecommand Tags for Setpoints in AUTTOM Control Panels**
N1 or B1	Not used
N2 or B2	Not used
N3 or B3	Not used
N4 or B4	Not used

In the **Device** parameter of these Tags, inform the identification data of the Control Panels using the format **ID:XXXXX:Rn:Tn**, as described next:

- **ID**: Identifier of a Modbus slave
- **XXXXX**: Type of Control Panel, according to the next values:
  - **CT850**: Ictel CT850 Control Panel
  - **AUTTOM_R1**: Radio 1 Auttom Control Panel
  - **AUTTOM_R2**: Radio 2 Auttom Control Panel
- **Rn**: Number of a Remote Station
- **Tn**: Type of Remote Station

The next tables contains the telecommands that can be used to interact with Control Panels, by using the **Item** parameter.

Telecommand Tags for Setpoints in CT850 Control Panels

TELECOMMAND	ITEM	DESCRIPTION	OPERATION
Writing request for a command Setpoint	Vn:SP_COMMAND	Sends a Setpoint telecommand to the Remote Station indicated by **Vn**, in the format **Vn = SetPointCount * 1000 + SetPointIndex**. For example, if the Control Panel contains 5 (five) Setpoints and users must change the third Setpoint, this telecommand is equal to **Vn = 5 * 1000 + 3 -> Vn = 5003**. For **Vn:SP_COMMAND** and **Vn:SP_BLOQ** requests, the writing value of this Tag represents the new Setpoint value to send to the Control Panel. For querying requests of Setpoints **Vn:SP_QUERY_COMMAND** and **Vn:SP_QUERY_BLOQ**, writing to this Tag can have any value. Use the respective **Vn:SP_RESPONSE** or **Vn:SP_RESP_BLOQ** Tag to check whether the processing of this telecommand was completed successfully or failed	Write-only
Querying request for a command Setpoint	Vn:SP_QUERY_COMMAND		
Writing request for a blocking Setpoint	Vn:SP_BLOQ		
Querying request for a blocking Setpoint	Vn:SP_QUERY_BLOQ		

TELECOMMAND	ITEM	DESCRIPTION	OPERATION
Reading the response of a writing or querying request of a command Setpoint	Vn:SP_RESPONSE	Reading Tag used to monitor the result of the last Setpoint telecommand sent to the Remote Station indicated by **Vn**, in the format **Vn = SetPointCount * 1000 + SetPointIndex**. For example, if the Control Panel contains 5 (five) Setpoints and users must change the third Setpoint, this telecommand is equal to **Vn = 5 * 1000 + 3 -> Vn = 5003**. If successful, this Tag returns a 32-bit integer, with the upper bits set to 0 (zero) and the lower bits containing the value written by the **Vn:SP_COMMAND** Tag. In case of failure, this Tag returns a 32-bit integer, with the upper bits containing the Control Panel error and the lower bits containing the value written by the **Vn:SP_COMMAND** Tag	Read-only
Reading the response of a writing or querying request of a blocking Setpoint	Vn:SP_RESP_BLOQ		
Writing request for an Analog Setpoint	Vn:SP_ANALOG	Sends an Analog Setpoint telecommand to the Remote Station indicated by **Vn**, in the format **Vn = SetPointCount * 1000 + 2**. This analog Setpoint has a fixed position at "SetPoint 2". For example, if the Control Panel contains 5 (five) Setpoints, this telecommand is equal to **Vn = 5 * 1000 + 2 -> Vn = 5002**. Use the **Vn:SP_RESP_ANALOG** Tag to check whether the processing of this telecommand was completed successfully or failed	Write-only

TELECOMMAND	ITEM	DESCRIPTION	OPERATION
Reading the response of a writing request of an Analog Setpoint	Vn:SP_RESP_ANALOG	Reading Tag used to monitor the result of the last Analog Setpoint telecommand sent to the Remote Station indicated by **Vn**, in the format **Vn = SetPointCount * 1000 + 2**. This Analog SetPoint has a fixed position at "SetPoint 2". For example, if the Control Panel contains 5 (five) Setpoints, this telecommand is equal to **Vn = 5 * 1000 + 2 -> Vn = 5002**. If successful, this Tag returns a 32-bit integer, with the upper bits set to 0 (zero) and the lower bits containing the value written by the **Vn:SP_ANALOG** Tag. In case of failure, this Tag returns a 32-bit integer, with the upper bits containing the Control Panel error and the lower bits containing the value written by the **Vn:SP_ANALOG** Tag	Read-only

Usage Example

Writing a Setpoint telecommand for the **CT850** Control Panel with identifier equal to 1 (one, CT850), Remote Station equal to 3 (three, R3), Type of Remote Station equal to 2 (two, T2), with a group of 5 Setpoints, to change Setpoint 3 (three, V5003):

- **Tag.ParamDevice**: 1:CT850:R3:T2
- **Tag.ParamItem**: V5003:SP_COMMAND
- **Writing value of this Tag**: 1234 (value of the Setpoint)

Return of this telecommand:

- **Tag.ParamDevice**: 1:CT850:R3:T2
- **Tag.ParamItem**: V5003:SP_RESPONSE
- **Reading value of this Tag in case of success**: 1234, that is, a 32-bit integer 0x000004D2, with the upper bits equal to 0 (zero) and the lower bits equal to 1234
- **Reading value of this Tag in case of failure**: 4850898, that is, a 32-bit integer 0x004A04D2, with the upper bits equal to 74 and the lower bits equal to 1234, in which the upper bits receive the error code 0x004A from the Control Panel

Telecommand Tags for Setpoints in AUTTOM Control Panels

TELECOMMAND	ITEM	DESCRIPTION	OPERATION
Writing request for a command Setpoint	Vn:SP_COMMAND:Mn	Sends a Setpoint telecommand to the Remote Station indicated by **Vn**, in the format **Vn = SetPointCount * 1000 + SetPointIndex**. For example, if the Control Panel contains 5 (five) Setpoints and users must change the third Setpoint, this telecommand is equal to **Vn = 5 * 1000 + 3 -> Vn = 5003**. Indicate in **Mn** the initial position of the Modbus memory linked to the group of Setpoints of this telecommand. For **Vn:SP_COMMAND:Mn** and **Vn:SP_BLOQ:Mn** requests, the writing value of this Tag represents the new Setpoint value to send to the Control Panel. For querying requests of Setpoints **Vn:SP_QUERY_COMMAND:Mn** and **Vn:SP_QUERY_BLOQ:Mn**, writing to this Tag can have any value. Use the **Vn:SP_RESPONSE:Mn** Tag to check whether the processing of this telecommand was completed successfully or failed	Write-only
Querying request for a command Setpoint	Vn:SP_QUERY_COMMAND:Mn		
Writing request for a blocking Setpoint	Vn:SP_BLOQ:Mn		
Querying request for a blocking Setpoint	Vn:SP_QUERY_BLOQ:Mn		

TELECOMMAND	ITEM	DESCRIPTION	OPERATION
Reading the response of a writing or querying request of a command Setpoint	Vn:SP_RESPONSE:Mn	Reading Tag used to monitor the result of the last Setpoint telecommand sent to the Remote Station indicated by **Vn**, in the format **Vn = SetPointCount * 1000 + SetPointIndex**. For example, if the Control Panel contains 5 (five) Setpoints and users must change the third Setpoint, this telecommand is equal to **Vn = 5 * 1000 + 3 -> Vn = 5003**. Indicate in **Mn** the initial position of the Modbus memory linked to the group of Setpoints of this telecommand. If successful, this Tag returns a 32-bit integer, with the upper bits set to 0 (zero) and the lower bits containing the value written by the **Vn:SP_COMMAND:Mn** Tag. In case of failure, this Tag returns a 32-bit integer, with the upper bits containing the Control Panel error and the lower bits containing the value written by the **Vn:SP_COMMAND:Mn** Tag	Read-only
Reading the response of a writing or querying request of a blocking Setpoint	Vn:SP_RESP_BLOQ:Mn		

Usage Example

Writing a Setpoint telecommand to the **AUTTOM** Control Panel with an identifier equal to 1 of Radio 1 (one, AUTTOM_R1), Remote Station equal to 2 (two, R2), Type of Remote Station equal to 2 (two, T2), with a group of 4 (four) Setpoints, to change Setpoint 2 (two, V4002), linked to Modbus address 40035 (M40035):

- **Tag.ParamDevice**: 1:AUTTOM_R1:R2:T2
- **Tag.ParamItem**: V4002:SP_COMMAND:M40035
- **Writing value of this Tag**: 1234 (value of the Setpoint)

Return of this telecommand:

- **Tag.ParamDevice**: 1:AUTTOM_R1:R2:T2
- **Tag.ParamItem**: V4002:SP_RESPONSE:M40035
- **Reading value of this Tag in case of success**: 1234, that is, a 32-bit integer 0x000004D2, with upper bits equal to 0 (zero) and lower bits equal to 1234
- **Reading value of this Tag in case of failure**: 4850898, that is, a 32-bit integer 0x004A04D2, with upper bits equal to 74 and lower bits equal to 1234, in which the upper bits receive the error code 0x004A from the Control Panel

In the case of the **AUTTOM** Control Panel, the writing value of a Setpoint is always stored at the Modbus memory address linked to that Setpoint. In the previous example, in case of success, the value 1234 is stored in the Modbus memory position 40035 (M40035).

Peripherals

Use PLC Tags to manipulate specific telecommands from Control Panels, using the parameters indicated on the next table.

Parameters for Control Panels

Device	Parameters of a device, in the format **ID:XXXXX:Rn:Tn**
Item	Parameters of a telecommand. For more information, please check table **Telecommand Tags for peripherals**
N1 or B1	Not used
N2 or B2	Not used
N3 or B3	Not used
N4 or B4	Not used

In the **Device** parameter of these Tags, inform the identification data of the Control Panels using the format **ID:XXXXX:Rn:Tn**, as described next:

- **ID**: Identifier of a Modbus slave
- **XXXXX**: Type of Control Panel, according to the next values:
  - **CT850**: Ictel CT850 Control Panel
  - **AUTTOM_R1**: Radio 1 Auttom Control Panel
  - **AUTTOM_R2**: Radio 2 Auttom Control Panel
- **Rn**: Number of a Remote Station
- **Tn**: Type of Remote Station

The next table contains the telecommands that can be used to interact with Control Panels, by using the **Item** parameter.

Telecommand Tags for peripherals

TELECOMMAND	ITEM	DESCRIPTION	OPERATION
Writing request on peripherals	Vn:PERIPHERAL:Mn[.DataType]	Sends a data telecommand to a peripheral linked to the Control Panel, in which **Vn** corresponds to the index (identifier) of this peripheral, **Mn** corresponds to the memory position of this peripheral, and **DataType** corresponds to data type **.u32** or **.i32** to send 32-bit integers. This parameter is optional and, if not specified, 16-bit integer values are sent. For **Vn:PERIPHERAL:Mn[.DataType]** requests, the writing value of this Tag represents the new value to send to a peripheral. For **Vn:QUERY_PERIPHERAL:Mn[.DataType]** querying requests, writing to this Tag can have any value. Use the respective **Vn:RESP_PERIPHERAL:Mn[.DataType]** Tag to check whether the processing of this telecommand was completed successfully or failed	Write-only
Querying request on peripherals	Vn:QUERY_PERIPHERAL:Mn[.DataType]		
Reading the response to a writing or querying request on peripherals	Vn:RESP_PERIPHERAL:Mn[.DataType]	Reading Tag used to monitor the result of the last data telecommand sent to the respective peripheral indicated by **Vn** in the memory position **Mn** and, finally, the data type **[.DataType]**, if used. In case of success, this Tag returns a 32-bit integer, with the upper bits set to 0 (zero) and the lower bits containing the value written by the **Vn:PERIPHERAL:Mn[.DataType]** Tag. In case of failure, this Tag returns a 32-bit integer, with the upper bits containing the error from the Control Panel and the lower bits containing the value written by the **Vn:PERIPHERAL:Mn[.DataType]** Tag	Read-only

Usage Example

Writing a telecommand for the **CT850** Control Panel with identifier equal to 1 (one, CT850), Remote Station equal to 3 (three, R3), Type of Remote Station equal to 2 (two, T2), to send the value 1234 to peripheral 1 (one, V1), in memory position 6 (six, M6):

- **Tag.ParamDevice**: 1:CT850:R3:T2
- **Tag.ParamItem**: V1:PERIPHERAL:M6
- **Writing value of this Tag**: 1234 (value sent to memory six of peripheral one)

Return of this telecommand:

- **Tag.ParamDevice**: 1:CT850:R3:T2
- **Tag.ParamItem**: V1:RESP_PERIPHERAL:M6
- **Reading value of this Tag in case of success**: 1234, that is, a 32-bit integer 0x000004D2, with upper bits equal to 0 (zero) and lower bits equal to 1234
- **Reading value of this Tag in case of failure**: 7603410, that is, a 32-bit integer 0x007404D2, with upper bits equal to 116 and lower bits equal to 1234, in which the upper bits receive the error code 0x0074 from the Control Panel

Modbus

Use PLC or Block Tags to manipulate Modbus registers, using the parameters on the next table.

Parameters for Modbus Tags

Device	Identifier of a Modbus slave
Item	Modbus parameters, in the format **XXmmmmm[.tt][.ss]**
N1 or B1	Not used
N2 or B2	Not used
N3 or B3	Not used
N4 or B4	Not used

In the **Item** parameter of these Tags, inform the parameters of the Modbus Tag using the format **XXmmmmm[.tt][.ss]**, as described next:

- The **XX** parameter represents the type of Modbus command (*FCode*):
  - **cl**: Coils | Bits | Read 01 / Write 15
  - **di**: Discrete Input (read only) | Bits | Read 02
  - **hr**: Holding Register | Words | Read 03 / Write 16
  - **ir**: Input Register (read only) | Words | Read 04
  - **scl**: Single Coil | Bit | Read 01 / Write 05
  - **shr**: Single Holding Register | Word | Read 03 / Write 06
- The **mmmmm** parameter indicates a Modbus memory address
- The **.tt** parameter is optional and indicates the type of interpretation applied to data. If not informed, data is interpreted as a **Word** data type:
  - **.u16**: Word
  - **.i16**: 16-bit integer
  - **.u32**: DWord
  - **.i32**: 32-bit integer
  - **.f**: 32-bit floating point
  - **.d**: 64-bit double
- The **.ss** parameter is optional and indicate the byte order applied to data. If not informed, the byte order applied is **.msb** (by7 by6 by5 by4 by3 by2 by1 by0):
  - **.sb**: Swap Byte (by6 by7 by4 by5 by2 by3 by0 by1)
  - **.sw**: Swap Word (by5 by4 by7 by6 by1 by0 by3 by2)
  - **.swsb**: Swap Word + Byte (by4 by5 by6 by7 by0 by1 by2 by3)

- **.sdw**: Swap DWord (by3 by2 by1 by0 by7 by6 by5 by4)
- **.sdwsb**: Swap DWord + Byte (by2 by3 by0 by1 by6 by7 by4 by5)
- **.sdwsw**: Swap DWord + Word (by1 by0 by3 by2 by5 by4 by7 by6)
- **.sdwswsb**: Swap DWord + Word + Byte (by0 by1 by2 by3 by4 by5 by6 by7)

Usage Examples

Reading a Holding Register at memory position 120:

- **Tag.ParamDevice**: 1 (one)
- **Tag.ParamItem**: hr120

Reading a Holding Register at memory position 120 with inverted bytes (*Swap Byte*):

- **Tag.ParamDevice**: 1 (one)
- **Tag.ParamItem**: hr120.sb

Reading a Holding Register at memory position 120 with inverted bytes (*Swap Byte*) and a signed integer (*int16*):

- **Tag.ParamDevice**: 1 (one)
- **Tag.ParamItem**: hr120.i16.sb

Documentation of I/O Interfaces

This section contains the documentation of I/O Interfaces referring to **CT850** 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.

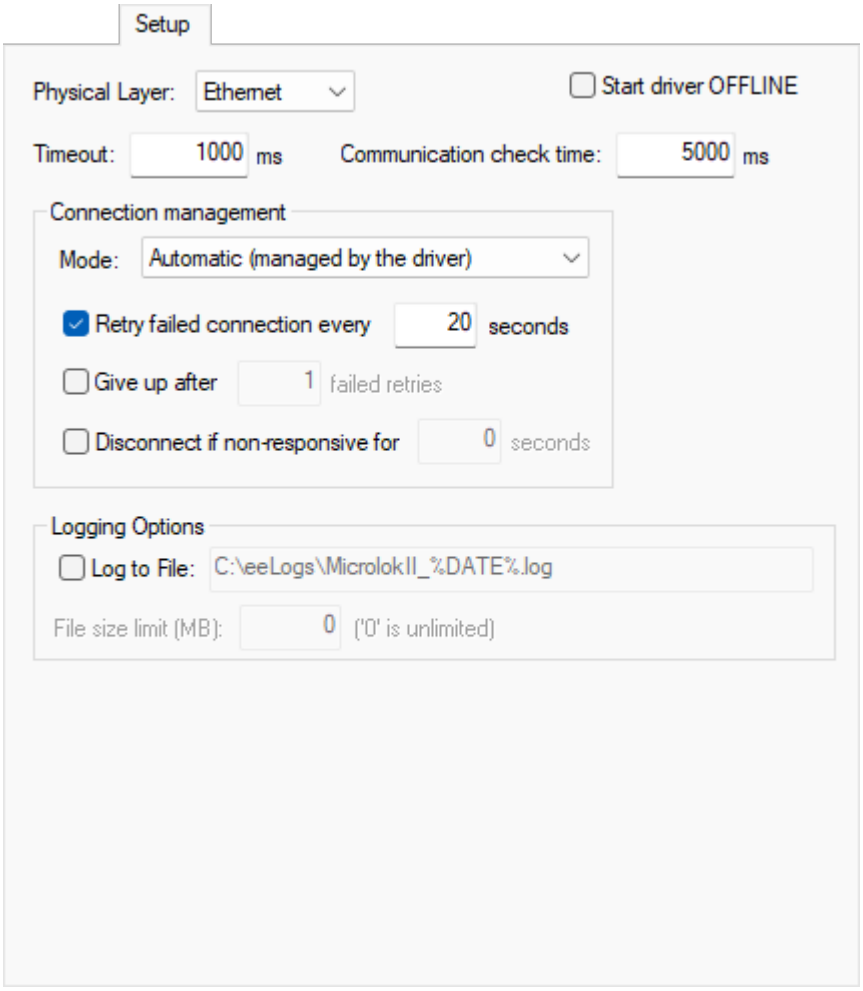
Configuration Dialog Box

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

OPTION	DESCRIPTION
Start driver OFFLINE	Select this option so that a Driver starts in **Offline** mode or stopped. This means that the I/O interface is not created until this Driver is configured to **Online** mode by using a Tag in an application. This mode enables a dynamic configuration of an I/O interface at run time

Options on the Connection management group

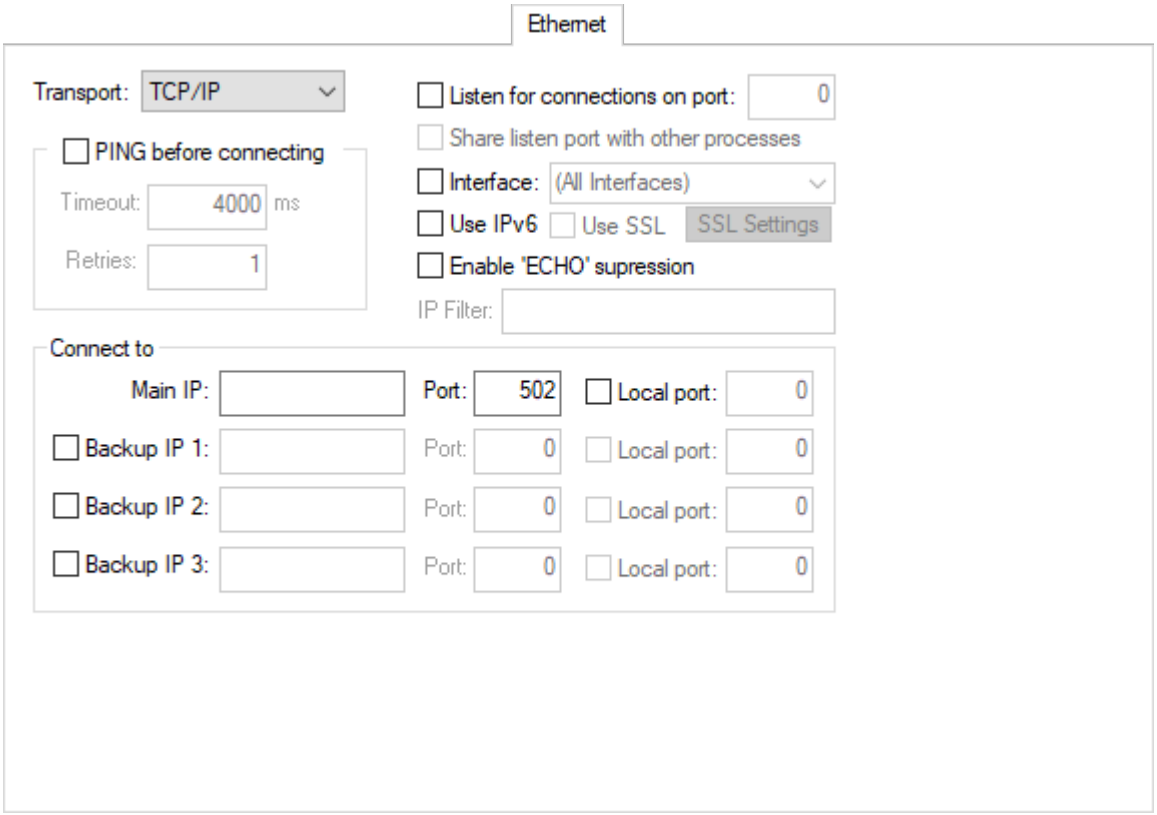
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection
Retry failed connection every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

Options on the Logging Options group

OPTION	DESCRIPTION
Log to File	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

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

OPTION	DESCRIPTION
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 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

General Configurations

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

I/O Tags

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

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

IO.CommunicationStatus

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

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

IO.IOKitEvent

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

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

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

NOTE

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

IO.PhysicalLayerStatus

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

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

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

IO.SetConfigurationParameters

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

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

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

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

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

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

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

A variation of the previous example uses a bidimensional array.

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

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

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

IO.WorkOnline

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

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

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

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

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

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

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

IMPORTANT

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

Properties

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

IO.ConnectionMode

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

IO.GiveUpEnable

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

IO.GiveUpTries

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

IO.InactivityEnable

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

IO.InactivityPeriodSec

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

IO.RecoverEnable

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

IO.RecoverPeriodSec

Delay time between two connection attempts, in seconds.

NOTE

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

IO.StartOffline

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

NOTE

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

IO.TimeoutMs

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

IO.Type

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

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

Statistical Configuration

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

I/O Tags

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

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

IO.Stats.Partial.BytesRecv

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

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

IO.Stats.Partial.BytesSent

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

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

IO.Stats.Partial.TimeConnectedSeconds

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

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

IO.Stats.Partial.TimeDisconnectedSeconds

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

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

IO.Stats.Total.BytesRecv

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

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

IO.Stats.Total.BytesSent

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

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

IO.Stats.Total.ConnectionCount

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

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

IO.Stats.Total.TimeConnectedSeconds

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

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

IO.Stats.Total.TimeDisconnectedSeconds

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

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

Properties

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

Ethernet Interface Configuration

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

I/O Tags

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

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

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

IO.Ethernet.IPSelect

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

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

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

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

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

IO.Ethernet.IPSwitch

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

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

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

IO.Ethernet.SocketState

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

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

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

Properties

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

NOTE

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

IO.Ethernet.AcceptConnection

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

IO.Ethernet.BackupEnable[2,3]

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

IO.Ethernet.BackupIP[2,3]

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

IO.Ethernet.BackupLocalPort[2,3]

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

IO.Ethernet.BackupLocalPortEnable[2,3]

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

IO.Ethernet.BackupPort[2,3]

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

IO.Ethernet.IPFilter

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

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

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

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

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

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

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

IO.Ethernet.ListenIP

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

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

IO.Ethernet.MainIP

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).

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
1.0.4	01/26/2026	C. Mello	• Added the **SP_ANALOG** telecommand to perform writings with analog Setpoints on **CT850** Control Panels (*Case 39332*).
1.0.3	12/30/2025	C. Mello	• Adjustments to **AUTTOM** Control Panels to follow the redirection performed by a legacy system when sending commands via Radio 2 (*Case 39277*).
1.0.2	09/30/2025	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 37978*).
		A. Fetzner	• Fixed the feedback of a command in **AUTTOM** Control Panels, which had its position reversed (*Case 38143*).
		C. Mello	• Added physical communication failures to the return codes of a Control Panel (*Case 38615*). • Added a buffering to individual Setpoint writings (*Case 37716*).
1.0.1	04/11/2025	C. Mello	• Initial version of this Driver (*Case 37522*).

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)