Sunix EAZINet Driver

File Name	EAZINet.dll
Manufacturer	SUNIX Group
Devices	Sunix data acquisition devices
Protocol	Sunix DevicePort SDK
Version	1.0.2
Last Update	06/22/2026
Platform	Win32
Dependencies	IOKit version 2.0 or later and Sunix DevicePort (IO_DLL.dll)
Superblock Readings	No
Level	0

Introduction

The Sunix EAZINet Driver was developed together with Sunix DevicePort SDK protocol to allow communication between **Elipse SCADA**, **Elipse E3**, **Elipse Power**, or **Elipse Water** applications and data acquisition devices by SUNIX Group.

Preparing a Device

For data acquisition devices by SUNIX Group to be recognized by a computer through EAZINet, users must install **SUNIX DevicePort Advanced** system driver, provided by that manufacturer.

Tag Reference

All Tags of this Driver must be referenced by the **ParamDevice** and **ParamItem** properties or by the **[N/B]** parameters, according to the next table.

ParamDevice property (Elipse E3, Elipse Power, or Elipse Water)	Specifies the number (identifier) of a device in **Number** format
ParamItem property (Elipse E3, Elipse Power, or Elipse Water)	Specifies a command in **Text** format
N1 or B1 parameter	Specifies the number (identifier) of a device in **Number** format
N2 or B2 parameter	Specifies a command in **Number** format
N3 or B3 parameter	Additional parameter, depends on the command used
N4 or B4 parameter	Additional parameter, depends on the command used

Optionally, and only available for **Elipse E3**, **Elipse Power**, or **Elipse Water** applications, users can use Tag Browser to click and drag Driver-defined Tags.

Configuration of Properties and Parameters in PLC and Block Tags

- **ParamDevice**: Property to identify a device by its number (identifier)
- **ParamItem**: Property that informs a command in **Text** format, such as "AO2/1" to indicate an action on the analog output of port 2 (two) and channel 1 (one) or "DIO3:4" to indicate an action on the digital input or output on port 3 (three) and bank 4 (four). More details on these commands are described in their specific topics
- **N1 or B1**: Parameter to identify a device by its number (identifier)
- **N2 or B2**: Parameter that informs a command in **Number** format, such as "300" to indicate an action on the analog output or "100" to indicate an action on the digital input or output. More details on these commands are described in their specific topics
- **N3 or B3**: Additional parameter that depends on the command used by the *N2 or B2* parameter
- **N4 or B4**: Additional parameter that depends on the command used by the *N2 or B2* parameter

NOTE

Applications designed in **Elipse SCADA** do not support **ParamDevice** and **ParamItem** properties, so that the configuration of commands must be performed only by using the **[N/B]** parameters of Tags.

Tags for Digital Inputs or Outputs

To manipulate digital input or output banks of data acquisition devices, PLC or Block Tags must be configured according to descriptive parameters on the **ParamItem** column or according to numerical parameters on **N2 or B2**, **N3 or B3**, and **N4 or B4** columns.

Tags related to Digital Inputs or Outputs

COMMAND	OPERATION	PARAMITEM	----	N2 OR B2	N3 OR B3	N4 OR B4
Value of digital inputs or outputs	When reading receives values from digital inputs or outputs and when writing sends values to digital outputs	**DIO**<*Port*>:<*Bank*>	or	100	<*Port*> × *100* + <*Bank*>	0 (zero)
Direction of digital inputs or outputs	When reading queries the current direction of digital ports and when writing defines a direction for digital ports. The direction of a digital port is defined with the values **0**: Defines a digital port as input or **1**: Defines a	**DIO_DIRECTION**<*Port*>:<*Bank*>	or	101		

COMMAND	OPERATION	PARAMITEM	----	N2 OR B2	N3 OR B3	N4 OR B4
	digital port as output					
Inversion of digital inputs	When reading queries the current status of the function to invert digital inputs and when writing enables or disables the function to invert the value of digital inputs. The function to invert the value of digital inputs is defined with the values **0**: Disables the inversion of digital inputs or **1**: Enables the inversion of digital inputs	**DI_INVERT**<*Port*>:<*Bank*>	or	102		
Clock of digital inputs or outputs	Write-only and defines the sample rate of digital ports based on the dividers **0**: 1, **1**: 50, **2**: 100, **3**: 2.5k, **4**: 5k, **5**: 50k, **6**: 100k, **7**: 2.5M, or **8**: 5M	**DIO_CLOCK**<*Port*>:<*Bank*>	or	103		

Reading or writing PLC Tags only acts on the configured port and bank, that is, *Value* refers to the user-configured <*Port*> and <*Bank*>.

Reading or writing Block Tags, on the other hand, acts sequentially on several ports and banks based on an initial combination of user-configured port and bank, considering the number of Elements defined for a Block Tag.

Assuming that a device contains a maximum number of 4 (four) banks per port, using a Block Tag with 8 (eight) Elements has the next result when processing a command from *Port1* and *Bank1*:

- **Element 1**: *Value1* (referring to <*Port1*> and <*Bank1*>)
- **Element 2**: *Value2* (referring to <*Port1*> and <*Bank2*>)
- **Element 3**: *Value3* (referring to <*Port1*> and <*Bank3*>)
- **Element 4**: *Value4* (referring to <*Port1*> and <*Bank4*>)

- **Element 5**: *Value5* (referring to <*Port2*> and <*Bank1*>)
- **Element 6**: *Value6* (referring to <*Port2*> and <*Bank2*>)
- **Element 7**: *Value7* (referring to <*Port2*> and <*Bank3*>)
- **Element 8**: *Value8* (referring to <*Port2*> and <*Bank4*>)

Handling Digital Inputs and Outputs with PLC Tags

Use PLC Tags to manipulate values from a specific port and bank, according to the examples described on the next table.

Examples of handling values with PLC Tags

COMMAND	PARAMITEM	----	N2	N3	N4	RESULT
Reading from port 2 and bank 1 of digital inputs	DIO2:1	or	100	201 (2 × 100 + 1)	0 (zero)	Reading from this Tag receives a value from *Port2* and *Bank1* of digital inputs, in **Number** format
Writing to port 1 and bank 3 of digital outputs	DIO1:3	or	100	103 (1 × 100 + 3)	0 (zero)	Writing to this Tag sends a user-defined value to *Port1* and *Bank3* of digital outputs, in **Number** format

Handling Digital Inputs and Output with Block Tags

Use Block Tags to manipulate values from a sequence of ports and banks, according to the examples described on the next table.

Examples of handling values with Block Tags

COMMAND	PARAMITEM	----	B2	B3	B4	ELEMENTS	RESULT
Reading 2 (two) banks of digital inputs starting with port 3 and bank 1	DIO3:1	or	100	301 (3 × 100 + 1)	0 (zero)	2 (two)	This Block Tag receives the value of 2 (two) banks in sequence of digital inputs from the initial combination of *Port3* and *Bank1*. The Elements are **1**: Value for *Port3* and *Bank1* and **2**:

COMMAND	PARAMITEM	----	B2	B3	B4	ELEMENTS	RESULT
							Value for *Port3* and *Bank2*
Writing to 6 (six) banks of digital outputs starting with port 4 and bank 7, assuming that port 4 contains no more than 8 (eight) banks	DIO4:7	or	100	407 (4 × 100 + 7)	0 (zero)	6 (six)	This Block Tag sends 6 (six) user-defined values to digital outputs from the initial combination of *Port4* and *Bank7*. The Elements are **1**: Value for *Port4* and *Bank7*, **2**: Value for *Port4* and *Bank8*, **3**: Value for *Port5* and *Bank1*, **4**: Value for *Port5* and *Bank2*, **5**: Value for *Port5* and *Bank3*, and **6**: Value for *Port5* and *Bank4*

Tags for Analog Inputs or Outputs

To manipulate digital input or output channels of data acquisition devices, PLC or Block Tags must be configured according to descriptive parameters on the **ParamItem** column or according to numerical parameters on **N2 or B2**, **N3 or B3**, and **N4 or B4** columns.

Tags related to Analog Inputs or Outputs

COMMAND	OPERATION	PARAMITEM	----	N2 OR B2	N3 OR B3	N4 OR B4
Analog inputs	Read-only and receives values from analog inputs, only via PLC Tag. The **Timestamp** property of this PLC Tag varies according to the sampling rate configured for the	**AI**<*Port*>/<*Channel*>	or	200	<*Port*>	<*Channel*>

COMMAND	OPERATION	PARAMITEM	----	N2 OR B2	N3 OR B3	N4 OR B4
	respective port or channel of an analog input					
On/off analog inputs	Write-only and enables or disables an analog input. The function to enable or disable an analog input is defined by the values **0**: Disables an analog input or **1**: Enables an analog input	**AI_ON_OFF**<*Port*>/<*Channel*>	or	201		
Analog input sampling rate	Write-only and defines a value for the sampling rate of the channels of an analog input	**AI_RATE**<*Port*>/<*Channel*>	or	202		
Analog input sampling rate divider	Write-only and defines a value for the divider of the sampling rate of the channel of an analog input	**AI_RATE_DIV**<*Port*>/<*Channel*>	or	203		
Analog input gain	Write-only and defines a value for the gain of the channel of an analog input	**AI_GAIN**<*Port*>/<*Channel*>	or	204		
Analog outputs	Write-only and sends values to analog outputs	**AO**<*Port*>/<*Channel*>	or	300		
On/off analog outputs	Write-only and enables or disables an analog output. The function to enable or disable an analog output is defined based on the values **0**: Disables an analog output or **1**: Enables	**AO_ON_OFF**<*Port*>/<*Channel*>	or	301		

COMMAND	OPERATION	PARAMITEM	----	N2 OR B2	N3 OR B3	N4 OR B4
	an analog output					
Analog output sampling rate	Write-only and defines a value for the sampling rate of the channel of an analog output	**AO_RATE**<*Port*>/<*Channel*>	or	302		
Analog output sampling rate divider	Write-only and defines a value for the divider of the sampling rate of the channel of an analog output	**AO_RATE_DIV** <*Port*> /<*Channel*>	or	303		
Analog output gain	Write-only and defines a value for the gain of the channel of an analog output	**AO_GAIN**<*Port*>/<*Channel*>	or	304		

Reading or writing PLC Tags only acts on the configured port and channel, that is, *Value* refers to the user-configured <*Port*> and <*Channel*>.

Reading or writing Block Tags, on the other hand, acts sequentially on several ports and channels based on the initial combination of user-configured port and channel, considering the number of Elements defined for a Block Tag.

Assuming that a device contains a maximum number of 4 (four) channels per port, using a Block Tag with 8 (eight) Elements has the next result when processing a command from *Port1* and *Channel1*:

- **Element 1**: *Value1* (referring to <*Port1*> and <*Channel1*>)
- **Element 2**: *Value2* (referring to <*Port1*> and <*Channel2*>)
- **Element 3**: *Value3* (referring to <*Port1*> and <*Channel3*>)
- **Element 4**: *Value4* (referring to <*Port1*> and <*Channel4*>)
- **Element 5**: *Value5* (referring to <*Port2*> and <*Channel1*>)
- **Element 6**: *Value6* (referring to <*Port2*> and <*Channel2*>)
- **Element 7**: *Value7* (referring to <*Port2*> and <*Channel3*>)
- **Element 8**: *Value8* (referring to <*Port2*> and <*Channel4*>)

NOTE

To read analog inputs, an application must only use PLC Tags. The **Timestamp** property of a Tag PLC varies according to the configured sample rate for the respective port or channel of an analog input.

Handling Analog Inputs and Outputs with PLC Tags

Use PLC Tags to receive values from a specific port and channel, according to the examples on the next table.

Examples of handling values with PLC Tags

COMMAND	PARAMITEM	----	N2	N3	N4	RESULT
Reading from port 2 and channel 1 of analog inputs	AI2/1	or	200	2 (two)	1 (one)	Reading from this Tag receives the value from *Port2* and *Channel1* of analog inputs, in **Number** format
Writing to port 1 and channel 3 of analog outputs	AO1/3	or	300	1 (one)	3 (three)	Writing to this Tag sends a user-defined value to *Port1* and *Channel3* of analog outputs, in **Number** format

Handling Analog Inputs and Outputs with Block Tags

Use Block Tags to receive values from a sequence of ports and channels, according to the examples on the next table.

Examples of handling values with Block Tags

COMMAND	PARAMITEM	----	B2	B3	B4	ELEMENTS	RESULT
Writing to define a sample rate in 2 (two) analog input channels, starting with port 3 and channel 1	AI_RATE3/1	or	202	3 (three)	1 (one)	2 (two)	This Block Tag sends 2 (two) user-defined values to analog inputs from the initial combination of *Port3* and *Channel1*. The Elements are **1**: Value for *Port3* and *Channel1* and **2**: Value for *Port3* and *Channel2*
Writing to 5 (five) analog output channels,	AO2/6	or	300	2 (two)	6 (six)	5 (five)	This Block Tag sends 5 (five) user-defined

COMMAND	PARAMITEM	----	B2	B3	B4	ELEMENTS	RESULT
starting with port 2 and channel 6, assuming that port 2 contains no more than 8 (eight) channels							values to analog outputs from the initial combination of *Port2* and *Channel6*. The Elements are **1**: Value for *Port2* and *Channel6*, **2**: Value for *Port2* and *Channel7*, **3**: Value for *Port2* and *Channel8*, **4**: Value for *Port3* and *Channel1*, and **5**: Value for *Port3* and *Channel2*

Documentation of I/O Interfaces

This section contains documentation about I/O Interfaces referring to **EAZINet** 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.

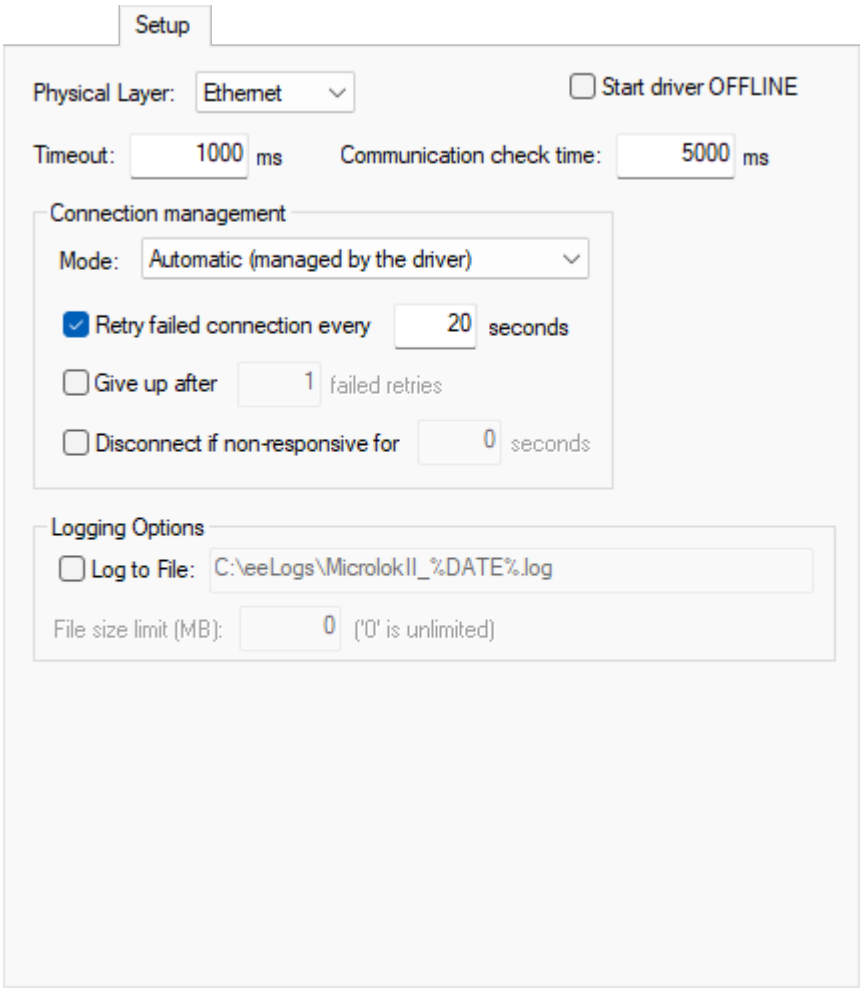
Configuration Dialog Box

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

OPTION	DESCRIPTION
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

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

9 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

9 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

9 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.

Driver's Revision History

VERSION	DATE	AUTHOR	COMMENTS
1.0.2	06/22/2026	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 38069*).
1.0.1	03/15/2018	C. Mello	• Initial version of this Driver (*Case 23403*).

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)