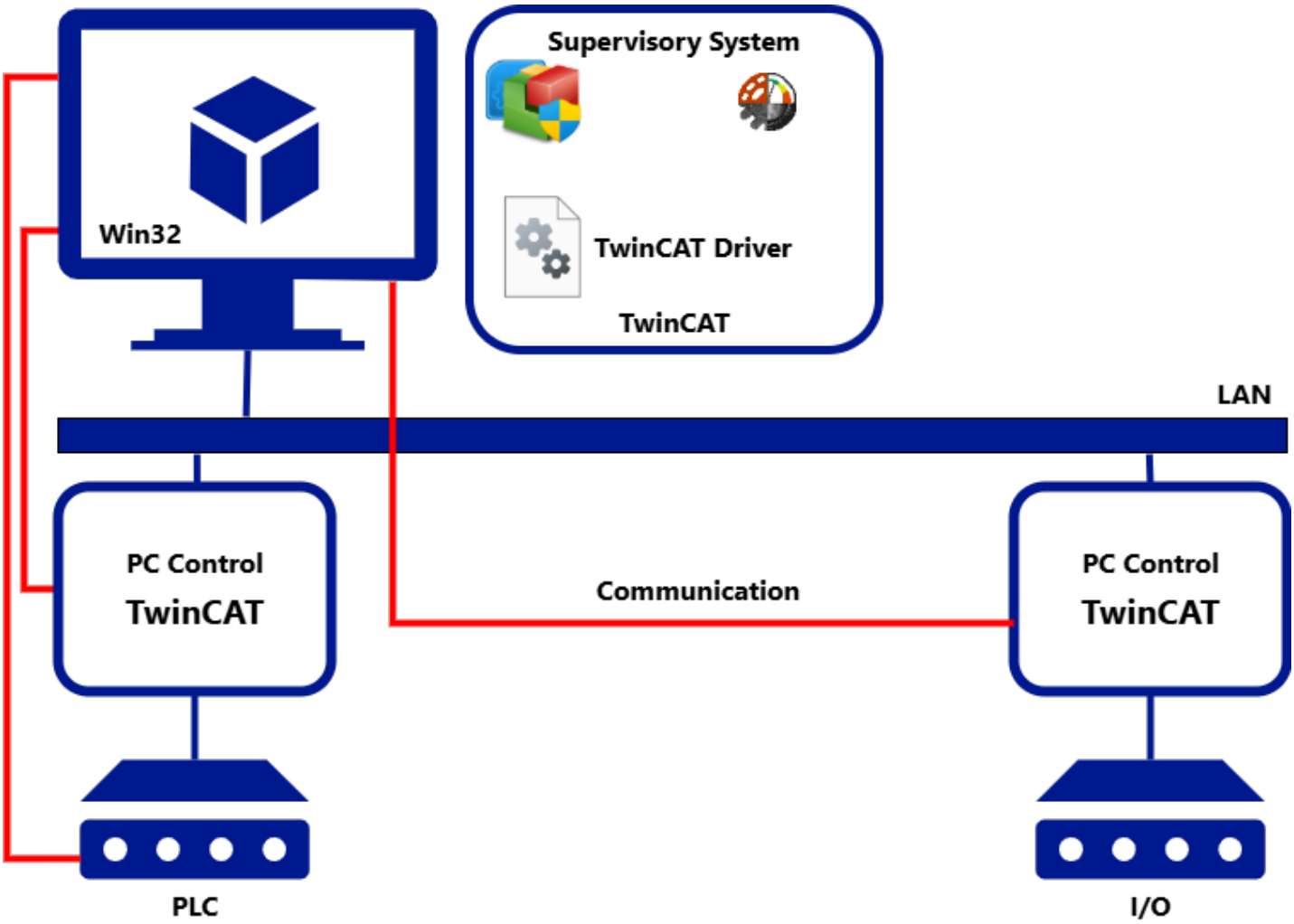
Beckhoff TwinCAT Driver

Filename	TwinCAT.dll
Manufacturer	Beckhoff
Devices	CX1000 Embedded PC and other Beckhoff devices communicating with TwinCAT ADS
Protocol	Automation Device Specification (ADS)
Version	2.0.7
Last Update	06/12/2025
Platform	Win32
Dependencies	TwinCAT System Control / TwinCAT PLC software and TcAdsDll.dll Library (Win32)
Superblock Readings	No
Level	0

Introduction

This is the Beckhoff TwinCAT Driver for communication between **Elipse Software** systems and Beckhoff devices. The architecture of a TwinCAT system allows individual software modules be handled as independent devices, that is, for each task there is a software module, client or server. System servers are *devices* executing as a software, whose operational behavior is exactly the same as a hardware device. Because of that they can be called *virtual* devices implemented via software. *Clients* are programs performing service requests from *servers*. Messages among these objects are exchanged using a consistent ADS (*Automation Device Specification*) interface via a *message router*. This entity manages and distributes all messages on the system and over TCP/IP connections. Message routers exist at each TwinCAT computer and at each Beckhoff BCxxxx bus controller. This allows TwinCAT servers and clients to exchange commands and data, send messages, and transfer status information, among other tasks.



Example of a topology TwinCAT / Elipse E3 or Elipse SCADA application

On a typical topology of a supervisory and control system, the supervisory application executing in an **Elipse Software** application works as a client, performing requests for reading or writing variables or memory from Beckhoff PLCs. These PLCs, by their turn, work as servers.

Device Configuration

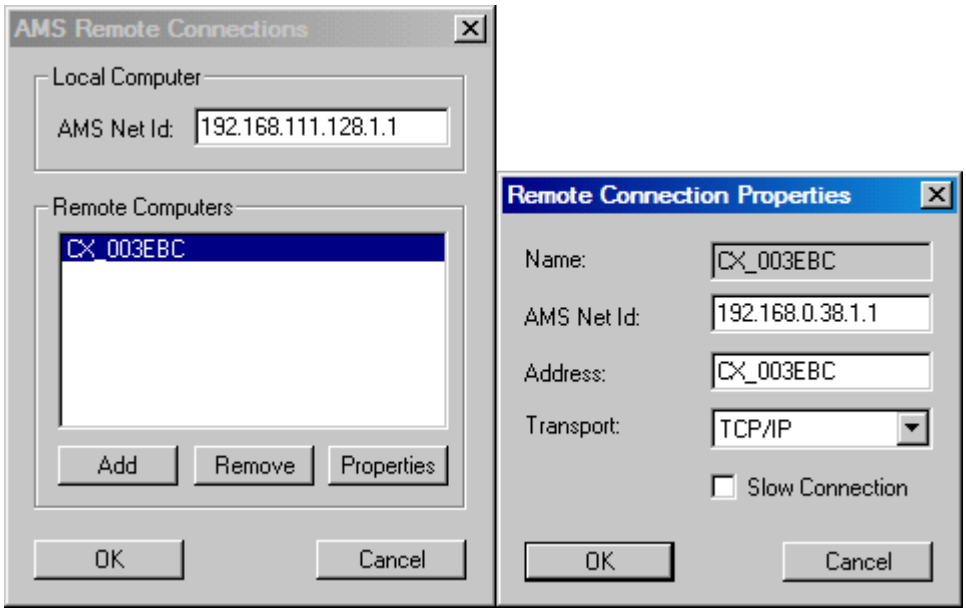
The requirement for a successful communication of a PLC with a supervisory application executing in an **Elipse Software** application is the presence of a *Ladder* program executing with its variables correctly configured in this Driver. This PLC must also have a known *AMS Net Id* and an available communication port, correctly configured in this Driver.

On the computer executing the supervisory application, on the other hand, users must configure the ADS routing by using a Beckhoff software.

When using **PLC CX1000 Embedded PC**, all remote connections can be configured using **AMS Remote Connections**. This software is on the installation directory of **TwinCAT System Manager**, on sub-folder **ADS Api/TcAdsDll**. This software presents the local *AMS Net Id* and a list of remote devices. Remote connections must be added by clicking **Add**, which opens a dialog box with the next options:

- **Name**: Name of a connection
- **AMS Net Id**: Net Id of a remote computer
- **Address**: IP address or host name
- **Transport**: Transport protocol, which is always "TCP/IP"

This software must be executed in **CX1000 PLC**, but this time configuring the computer executing the supervisory application as the remote computer.



Configuration of AMS Remote Connections application

In other devices, the routing task can be easily performed by using **TwinCAT System Manager**. In it, users can identify all ADS devices by scanning the whole network and, after creating a list of devices, selecting the items to add a routing. In this case, the routing is performed bidirectionally, that is, Computer » Device and Device » Computer, and not only on a single direction, as with **AMS Remote Connections**.

After finishing all ADS routing configurations, restart all configured devices. For more information about these applications and Beckhoff's TwinCAT PLCs, please check their documentation.

Driver Configuration

This Driver's **[P]** configuration parameters are not used. All configurations are performed on this Driver's extra configurations dialog box, on the **TwinCAT** and **TwinCAT Names** tabs.

NOTE

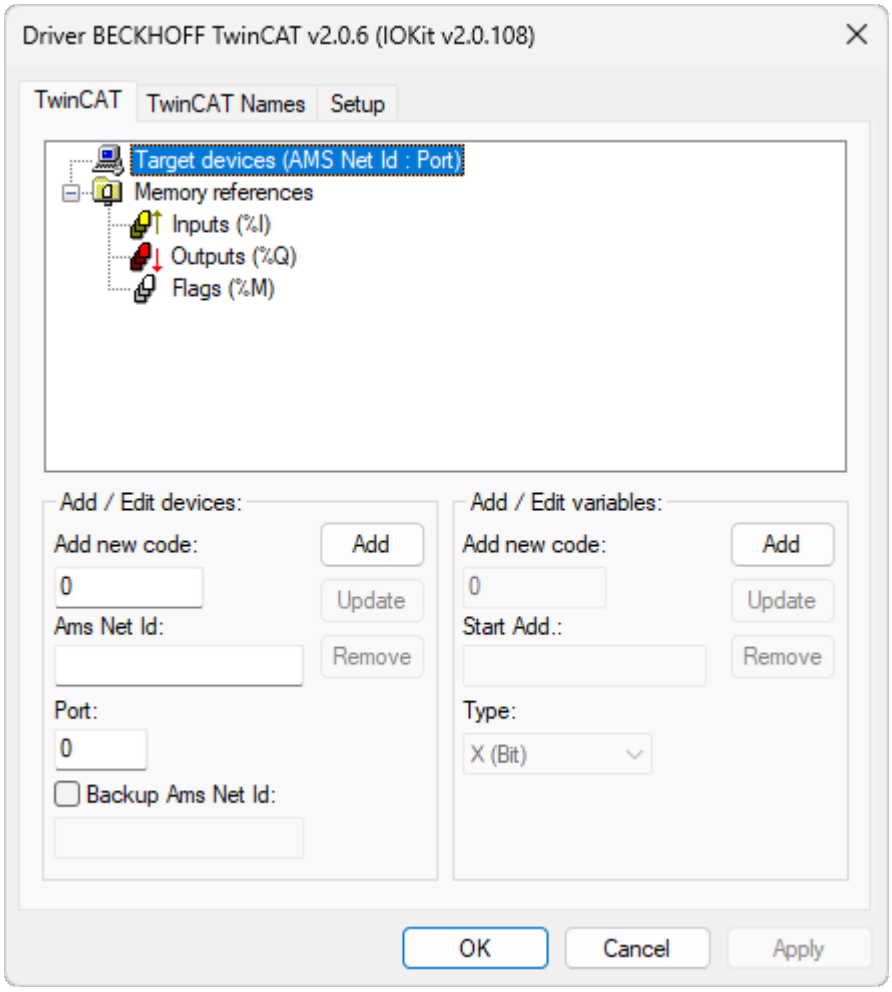
Elipse E3, **Elipse Power**, or **Elipse Water** allow configuring Tags by syntactical parameters and, when opting by this configuration mode, there is no need to fill the lists on the **TwinCAT** and **TwinCAT Names** tabs. These tabs are only used when configuring by numerical parameters and can be completely ignored when using syntactical parameters.

TwinCAT Tab

This tab displays a tree with two groups of items to fill in. The **Target Devices** item is used to insert connections with remote devices that communicate with this Driver, and users must configure at least one connection to establish a successful communication.

The **Memory references** item is used to add references to memory addresses, which are divided into physical inputs (**Inputs %I**), physical outputs (**Outputs %Q**), and general usage memories (**Flags %M**).

The next tables show instructions on how to fill in these groups of items.



TwinCAT tab

The available options on this tab are described on the next table.

Available options on the Target devices group

OPTION	DESCRIPTION
Add new code	Fill it in with an index number, a code between 0 (zero) and 255. In case there is already a connection with this number, it is automatically selected on the table and its values are loaded to the edition fields. This code is a reference to a connection with a PLC or any other Beckhoff device on Tags, that is, the index configured on the list is the one to indicate on Tags by the *N2/B2* parameter
Ams Net Id	Configure here the *Ams Net Id* of the main destination device. It must be formed by eight numbers separated by dots
Port	*Ams Net Id* port to which this Driver must connect
Backup Ams Net Id	Select the check box and configure here the *Ams Net Id* of the backup destination device, in case of a redundancy architecture. It must be formed by eight numbers separated by dots

Available options on the Memory references group

OPTION	DESCRIPTION
Add new code	Fill it in with an index number, a code between 0 (zero) and 65535. In case there is already a memory with this number, it is automatically selected on the list and its values are loaded to the edition fields. This code is a reference to the address of a variable, that is, the index configured on the list must be the one indicated on Tags for reading or writing variables by address by the *N3/B3* parameter
Start Add	Variable's initial address. It is only allowed to insert numbers in decimal notation. In case of a **Bit** data type, two numbers must be inserted and separated by a dot, the second number indicating the position of the bit to read, such as "100.1", that is, bit 1 (one) of memory address 100
Type	Memory format. Possible values are **X** (*Bit*), **B** (*Byte*), **W** (*Word*), **D** (*Double Word*), or **F** (*Float*). Properties of these formats can be checked on the table **Types of Variables**. The type of memory may also vary between **Physical input** (*%I*), **Physical output** (*%Q*), or **Internal memory** (*%M*), depending on the selected item on the tree

The options to manipulate data on these lists are the following:

- **Add**: Adds parameters
- **Update**: Changes parameters already listed
- **Remove**: Completely removes a row of parameters

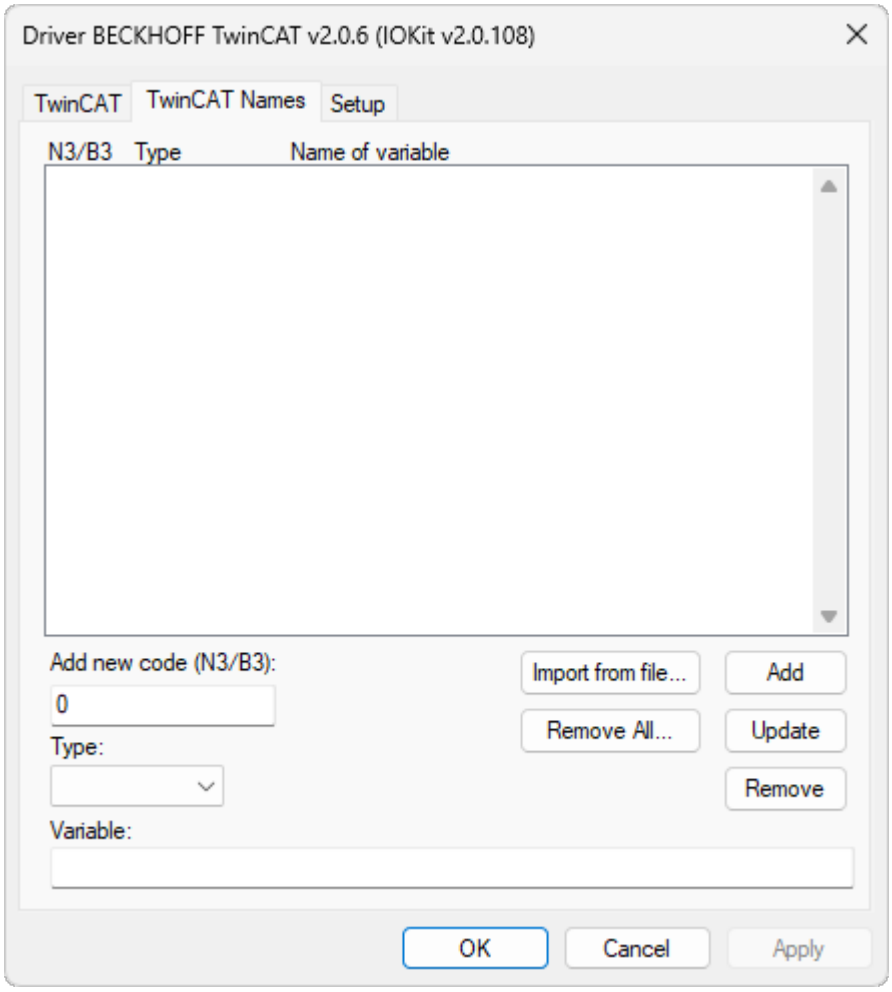
To add a reference to a destination device, select the **Target Devices** item, fill in the fields according to table **Available options on the Target devices group**, and then click **Add**.

To add a reference to a memory field, select the icon of a type (for a physical input, for a physical output, or for an internal memory), fill in the fields according to table **Available options on the Memory references group**, and then click **Add**.

To change or remove any of the existing references, select the item representing it and then click **Update** or **Remove**.

TwinCAT Names Tab

Similar to the **TwinCAT** tab, this tab presents a list of references for names of variables. Users must fill in this list to read or write variables with names and not numerical addresses.



TwinCAT Names tab

The available options on this tab are described on the next table.

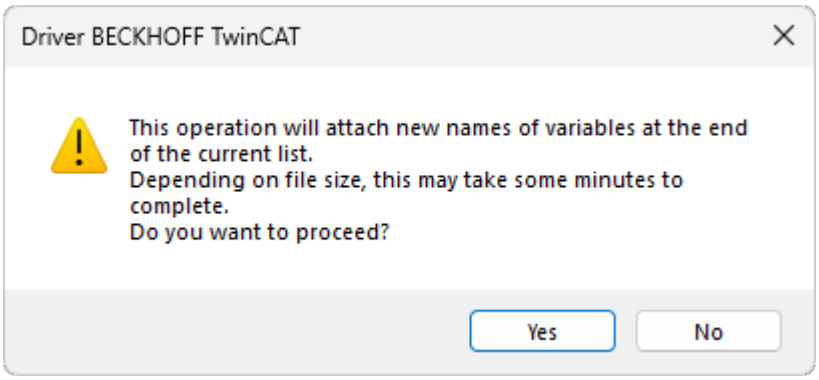
Available options on the TwinCAT Names tab

OPTION	DESCRIPTION
Add new code (N3/B3)	Fill it in with an index number, a code between 0 (zero) and 99999. In case there is already a variable with this number, it is automatically selected on the table and its values are loaded to the edition fields. This code is a reference to a name of a variable, that is, the index configured on this list must be indicated on Tags for reading or writing variables by name in *N3/B3* parameter
Type	Data type of a variable. All data types of variables are listed on the table **Types of Variables**
Variable	Name of a variable. The maximum size is 100 characters

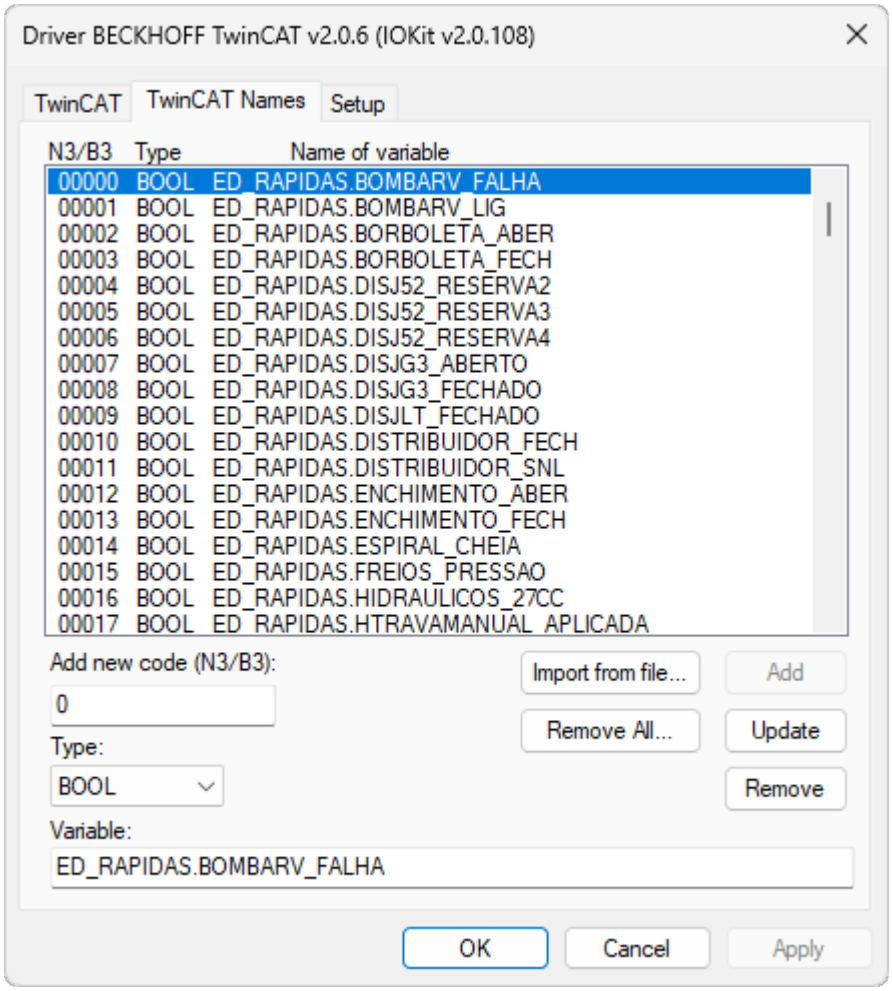
In addition to the options **Add**, **Update**, and **Remove** to manipulate the list, the **TwinCAT Names** tab also contains options to import and remove names of variables. The **Import from file** option opens a dialog box to select symbol files (SYM), which are files generated by the design software of the PLC's Ladder program, **TwinCAT PLC Control**. The operation to import a SYM file is very useful when a project works with several variables, by reducing the time spent on typing names of variables to configure this Driver.

To generate a SYM file with names of variables using the software **TwinCAT PLC Control**, users must enable all items on the dialog box opened by the **Configure symbol file** option on the **Options - Symbol configuration** menu. After that, execute the **Project - Rebuild all** option to generate a SYM file with all names of variables in a project.

To perform the import process of names of variables from a SYM file, click **Import from file** and select the file to import. By confirming this operation, the process of reading and converting this file starts and, depending on its size and the available resources on the computer, can be a time-consuming operation. In this case, a bar indicates the progress of the import process.



Import message



Result of the process of importing a SYM file

The **Remove all** option is used to remove all names of variables from the list. Click **Yes** to confirm this removal operation.

Other Configurations

On the **Setup** tab, users must select the **None** item on the **Physical Layer** option, indicating that no physical interface is directly triggered by this Driver. This is the default option when adding a new Driver object in an application.

Any configuration other than the ones mentioned until now are irrelevant and have no effect, so they must be disregarded and kept with their default values.

Click **OK** to save all configurations and close this Driver's properties window.

Types of variables

DATA TYPE	SIZE IN BITS	FORMAT	RANGE OF VALUES
BIT	1	Bit	0 (zero) or 1 (one)
BOOL	8	Boolean	FALSE or TRUE
BYTE	8	Byte	0 (zero) to 255
WORD	16	Word	0 (zero) to 65535
DWORD	32	Double Word	0 (zero) to 4294967295
SINT	8	Signed 8-bit integer	-128 to 127
USINT	8	Unsigned 8-bit integer	0 (zero) to 255
INT	16	Signed 16-bit integer	-32768 to 32767
UINT	16	Unsigned 16-bit integer	0 (zero) to 65535
DINT	32	Signed 32-bit integer	-2147483648 to 2147483647
UDINT	32	Unsigned 32-bit integer	0 (zero) to 4294967295
REAL or FLOAT	32	Single precision floating point in **IEEE 754** standard	-3.402823e+38 to 3.402823e+38
LREAL or DOUBLE	64	Double precision floating point in **IEEE 754** standard	-1.79769313486231e308 to 1.79769313486232e308
STRING	-	Text in **ASCII** format	Maximum of 255 characters
TIME	32	Time in milliseconds	0ms to 71582m47s295ms
TIME_OF_DAY (TOD)	32	Time of the day in milliseconds	00:00:00 to 1193:02:47.295
DATE	32	Current date	1970-01-01 to 2106-02-06
DATE_AND_TIME (DT)	32	Current date and time	1970-01-01-00:00 to 2106-02-06-06:28:15

Tag Reference

Tags in this Driver can be referenced in **Elipse E3**, **Elipse Power**, or **Elipse Water** using **Strings** in the **Device** and **Item** fields or else by **[N/B]** numerical parameters.

Configuration by Syntactical Parameters

Elipse E3, starting with version **2.0**, **Elipse Power**, or **Elipse Water** allow configuring Tags by syntactical parameters. Use the next syntax for each field:

- The **Device** field must obey the syntax **<Main Ams Net Id>:<Port Number> [{Backup Ams Net Id}]**:
  - **<Main Ams Net Id>** must be filled in with the *Net Id* of the destination device, which is composed of eight numbers separated by dots
  - **<Port number>** must be filled in with the port value of the *Ams Net Id* to which this Driver must connect

- The **{Backup Ams Net Id}** item is optional and if selected, it must also be composed of eight numbers separated by dots

- The **Item** field must obey a specific syntax for each Tag:
  - **<Variable name>:<Type>** for **Variables by Name**
  - **%<Area-Type><Starting address>** for **Variables by Address**
  - Fill in with the **String** "TwinCAT.ADSStatus" for the **homonym Tag**
  - Same for the **TwinCAT.NetIdSelect** Tag
  - Same for the **TwinCAT.NetIdSwitch** Tag

Configuration by Numerical Parameters (N/B)

When *N/B* parameters are used, Tags are referenced to commands by configuring the *N1/B1* parameter. The *N2/B2* parameter, in any Tag, is dedicated to the configuration of the destination terminal to which a command must be redirected. Fill it in with the corresponding numerical index that was assigned on the list of connections of the configuration dialog box, as explained on topic **Driver Configuration**.

NOTE
The *N/B* parameters not mentioned for filling are irrelevant when configuring Tags. It is recommended that these values remain in 0 (zero).

Variables by Name Tag

Reading and Writing

Configuration by numerical parameters

PARAMETER	VALUE
N1/B1	0 (zero)
N2/B2	Destination terminal
N3/B3	Name reference
N4/B4	0 (zero)

The *N3/B3* parameter must be configured with the index of one of the variables filled in on the list of extra parameters. To do so, please check the **TwinCAT Names** tab.

Configuration by syntactical parameters

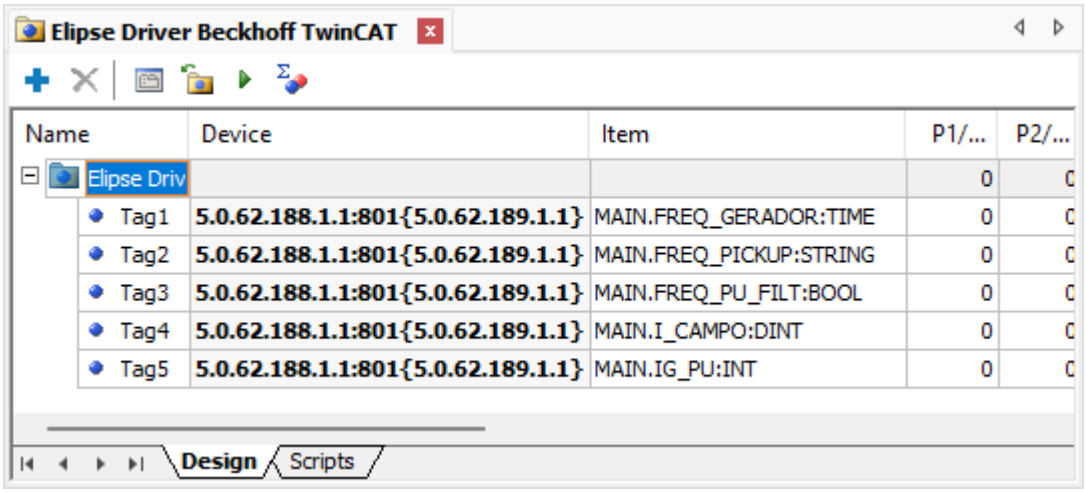
PARAMETER	VALUE
Device	<Main Ams Net Id>:<Port number>[{Backup Ams Net Id}]
Item	<Variable name>:<Type>

- **Value field**: This field displays the content of a variable, regardless of the data type it represents.

This Tag can be either an I/O Tag or a Block Tag. Using an I/O Tag or a Block Tag with one Element represents a simple variable. Use a Block Tag with more than one Element to represent an array. The **Strings** for data types are the following:

- **BOOL**
- **BYTE**
- **WORD**
- **DWORD**
- **SINT**
- **USINT**
- **INT**
- **UINT**
- **DINT**
- **UDINT**
- **REAL**
- **LREAL**
- **STRING**
- **TIME**
- **TOD**
- **DATE**
- **DT**

The meaning of these data type **Strings** can be checked on the table **Types of Variables**.



Example of syntactical configuration of a Variables by Name Tag

Variables by Address Tag

Reading and Writing

Configuration by numerical parameters

PARAMETER	VALUE
N1/B1	1 (one)
N2/B2	Destination terminal
N3/B3	Address reference
N4/B4	0 (zero)

The *N3/B3* parameter must be configured with the index of one of the variables added on the list of references to the memories of extra parameters. To do so, please check the **TwinCAT** tab.

Configuration by syntactical parameters

PARAMETER	VALUE
Dispositivo	<Main Ams Net Id>:<Port number>[{Backup Ams Net Id}]
Item	%<Area-Type><Initial address>

- **Value field**: This field displays the variable's content, regardless of the data type it represents.

This Tag can be either an I/O Tag or a Block Tag. Using an I/O Tag or a Block Tag with one Element represents a simple variable read at the referenced memory position. Use a Block Tag with more than one Element to represent reading sequential memory blocks, in which each Element reflects a different memory position, interleaved with its neighbor Element for the space occupied by the data type of that variable. This is useful when performing an optimized reading of many memory elements in a PLC close to each other.

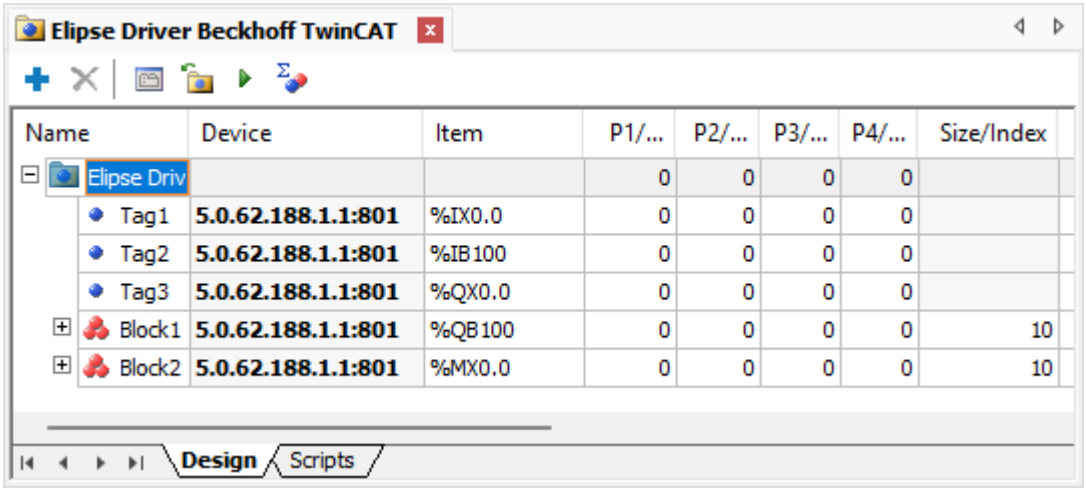
For example, memory fields **%MW100** and **%MW104** can be read at the same time in a Block Tag with three Elements. To do so, configure the starting address as 100. The first Element contains the value of **%MW100** and the third one contains the value of **%MW104**. This is already an advantage over reading these fields if they are separately referenced in I/O Tags, even by considering that **%MW102**, the field located between both fields, is read without need.

The exception is relative to memories with **X** data types (*Bit*), which can only be read or written by Tags with one Element, I/O or Block.

Area and data type characters

AREA AND DATA TYPE STRING	MEANING
IX	Individual physical input of a **Bit** data type
IB	Physical inputs interpreted as a **Byte** data type with 8 (eight) bits
IW	Physical inputs interpreted as a **Word** data type with 16 bits
ID	Physical inputs interpreted as a **Double Word** data type with 32 bits
IF	Physical inputs interpreted as a **Float** data type with 32 bits
QX	Individual physical output of a **Bit** data type
QB	Physical outputs interpreted as a **Byte** data type with 8 (eight) bits

AREA AND DATA TYPE STRING	MEANING
QW	Physical outputs interpreted as a **Word** data type with 16 bits
QD	Physical outputs interpreted as a **Double Word** data type with 32 bits
QF	Physical outputs interpreted as a **Float** data type with 32 bits
MX	Internal memory as a **Bit** data type
MB	Internal memory as a **Byte** data type with 8 (eight) bits
MW	Internal memory as a **Word** data type with 16 bits
MD	Internal memory as a **Double Word** data type with 32 bits
MF	Internal memory as a **Float** data type with 32 bits



Variables by Address Tags

ADSStatus Tag

Reading and Writing

Configuration by numerical parameters

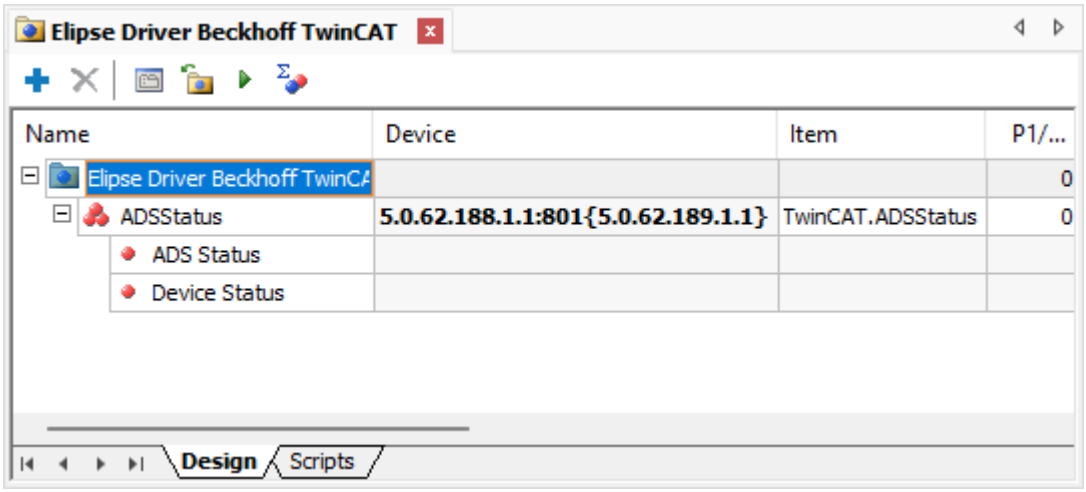
PARAMETER	VALUE
N1/B1	2 (two)
N2/B2	Destination terminal
N3/B3	0 (zero)
N4/B4	0 (zero)

Configuration by syntactical parameters

PARAMETER	VALUE
Device	<Main Ams Net Id>:<Port number>[{Backup Ams Net Id}]
Item	TwinCAT.ADSStatus

Reads or writes the **ADS** and **Device** statuses. It needs a Block Tag with two Elements:

- **Element 1**: ADS Status
- **Element 2**: Device Status



Example of syntactical configuration of an ADSStatus Tag

NetIdSelect Tag

Reading and Writing

Configuration by numerical parameters

PARAMETER	VALUE
N1/B1	10
N2/B2	Destination terminal
N3/B3	0 (zero)
N4/B4	0 (zero)

Configuration by syntactical parameters

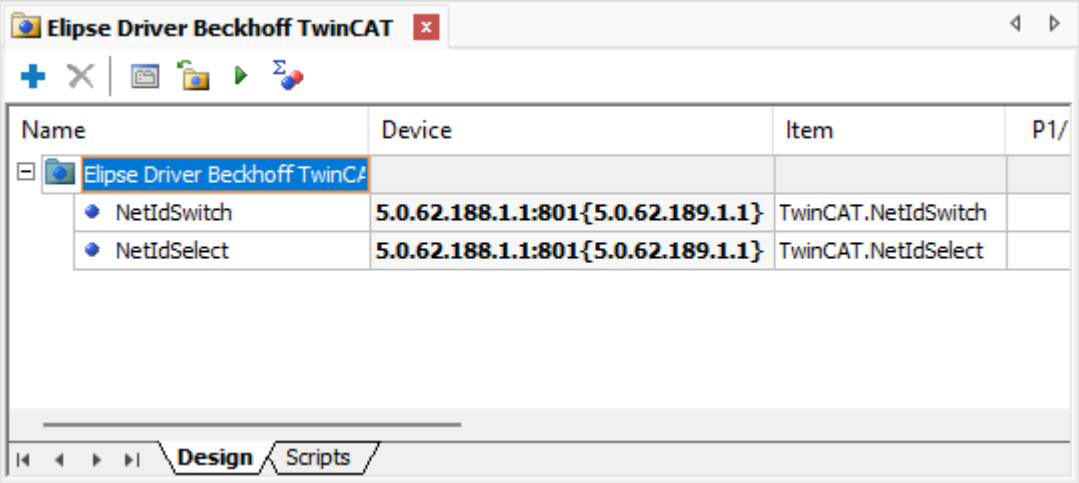
PARAMETER	VALUE
Device	<Main Ams Net Id>:<Port number>[{Backup Ams Net Id}]
Item	TwinCAT.NetIdSelect

- **Value field**: Indicates which *Ams Net Id* is the active one:
  - **0**: The main *Net Id* is selected (active)
  - **1**: The backup *Net Id* is selected (active)

If a connection is active, this Tag indicates which one of the two configured *Net Ids* is in use. If disconnected, this Tag indicates which one of the *Net Ids* is used first in the next connection attempt.

During the connection process, if the active *Net Id* is not available, then this Driver tries to connect with the other *Net Id*. If the connection to the alternate *Net Id* succeeds, then it is set as the active *Net Id*, that is, an automatic switching occurs.

To force a manual switch, write 0 (zero) or 1 (one) to this Tag. This forces a communication redirection to the other *Net Id* (**0**: Main *Net Id* or **1**: Backup *Net Id*) if this Driver is currently connected. If this Driver is disconnected, then it only sets the active *Net Id* for the next connection attempt.



Example of syntactical configuration of NetId Tags

NetIdSwitch Tag

Write-Only

Configuration by numerical parameters

PARAMETER	VALUE
N1/B1	11
N2/B2	Destination terminal
N3/B3	0 (zero)
N4/B4	0 (zero)

Configuration by syntactical parameters

PARAMETER	VALUE
Device	<Main Ams Net Id>:<Port number>[{Backup Ams Net Id}]
Item	TwinCAT.NetIdSwitch

Write any value to this Tag to send a command to this Driver and perform a manual switch of the destination *Ams Net Id*. If the main *Net Id* was active, then the backup *Ams Net Id* is activated and vice versa.

Support for Redundancy

Support for redundancy between two remote devices, one as main and another one as backup, is automatically selected every time that, for a given device, is also configured a backup device.

If there is a backup address configured, this Driver creates an exclusive execution thread to verify the status of a connection, which is periodically executed every second. If there is any failure on the main *Ams Net Id* connection, the

backup is automatically selected. Likewise, when the backup address is selected and its connection fails, the main *Ams Net Id* is selected again.

To manually control the connections with main and backup devices, use the **NetIdSelect** or **NetIdSwitch** Tags, as explained previously.

Documentation of I/O Interfaces

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

Configuration of a Driver

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

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

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

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

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

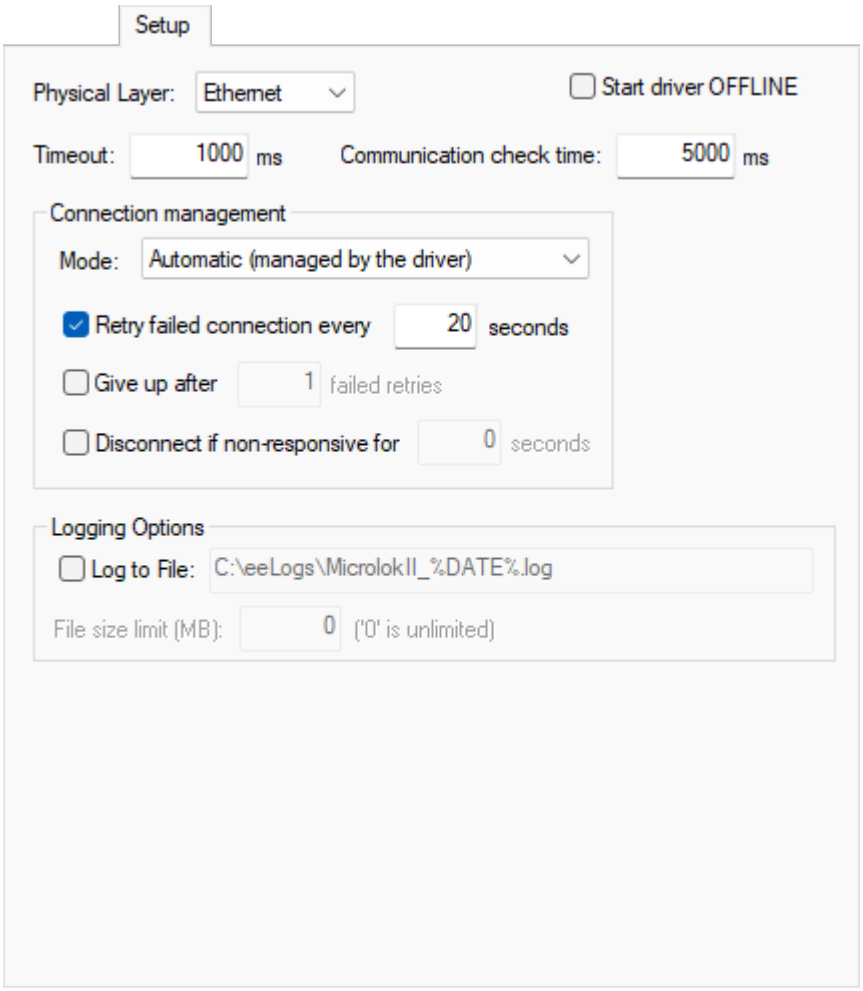
Configuration Dialog Box

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

Setup Tab

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

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



Setup tab

General options on the Setup tab

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

Options on the Connection management group

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

Options on the Logging Options group

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

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 Revision History

VERSION	DATE	AUTHOR	COMMENTS
2.0.7	06/12/2025	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and to Visual Studio 2022 (*Case 37496*).
2.0.6	08/09/2019	M. Ludwig	• Driver ported to Visual Studio 2017 (*Case 27293*).
2.0.5	08/25/2016	M. Ludwig	• Fixed a failure when reading variables with **String** data types (*Case 21390*).
2.0.4	08/16/2016	M. Ludwig	• Increased the maximum size of characters in **String** data types (*Case 21324*).
2.0.3	08/27/2014	M. Ludwig	• Fixed a failure when parsing variable names (*Case 16869*).
2.0.2	12/26/2013	M. Ludwig	• Driver ported to **IOKit** library version **2.0** (*Case 14630*). • Implemented redundancy and backup addresses (*Case 14841*). • Fixed an incorrect permission of configuring a variable with a blank name (*Case 14872*). • Implemented readings in callback functions (*Case 14876*). • Implemented an internal block that prevents selecting communication's physical layer (*Case 14982*). • Updated Beckhoff's **TcAdsDll** library (*Case 14998*). • Implemented a dynamic loading of **TcAdsDll** library (*Case 15036*).
1.3.1	08/10/2012	G. Taschetto	• Fixed a failure when performing an operation on a bit above seven (*Case 12989*).
1.2.1	12/03/2007	M. Ludwig	• Fixed a problem in the individual access to Block Elements, that is, **Array**-type variables (*Case 8734*). • Changed the limitation on the size of the list of

VERSION	DATE	AUTHOR	COMMENTS
			variables to avoid data loss (*Case 8906*).
1.1.1	12/13/2006	M. Ludwig	• Fixed a retention of invalid symbols when restarting a PLC (*Case 7690*).
1.0.1	09/14/2006	M. Ludwig	• First version of this Driver.

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

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

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

www.elipse.com.br

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

Gartner 2014 Cool Vendor

OPC FOUNDATION MEMBER

Gartner, Cool Vendors in Brazil 2014, April 2014.
Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability of fitness for a particular purpose.

Microsoft Partner
Gold Independent Software Vendor (ISV)