SIEMENS MProt (MPI/PPI/ISO-TCP) Driver

Filename	MProt.dll
Manufacturer	Siemens, Vipa
Devices	PLCs S7-200, S7-300, and S7-400 Siemens' PLCs series; Vipa's Speed7 and other devices compatible with any protocol of the Driver
Protocol	PPI and MPI (Serial); MPI encapsulated in Ethernet and ISO over TCP (RFC1006 or S7-TCP/IP on Ethernet interface)
Version	4.0.19
Last Update	08/05/2019
Platform	Win32
Dependencies	IOKit v2.00
Superblocks	Yes
Level	0

Introduction

The Siemens multi-protocol (M-Prot) Driver communicates with Siemens S7-200, S7-300, S7-400, and S7-1200 PLCs, as well as VIPA's Speed7 device using Siemens PPI, MPI, ISOTCP, and MPI protocols encapsulated in Ethernet MPI (IBHLink).

The **PPI** protocol must be used only with the S7-200 series, by using an RS232-PPI/MPI converter cable provided by Siemens.

The **MPI** protocol can be used with the S7-300 and S7-400 series using an RS232-PPI/MPI converter cable provided by Siemens, or also with VIPA's Speed7 series on the MPI port using a common RS-232 cable.

The **ISOTCP** protocol (which is also known as ISO over TCP, RFC1006, or S7-TCP/IP on several hardware vendor brochures) can be used with the Siemens S7-300 and S7-400 series by using a CP-3XX, CP-433, or CP-443 Ethernet card; for the S7-1200 model, and also for VIPA's Speed7 series, directly on the CPU's Ethernet port. For the S7-200 model, there is a special variation of the **ISOTCP** protocol for use with the CP-243 interface. This protocol is called **ISOTCP243**.

For PLCs without an Ethernet port, an alternative could be the Ethernet/MPI IBHLink converter provided by IBH Softec or Hilscher, which works on the FDL level. By using this converter, the advantage is a faster nominal speed, up to 187 kbps on an MPI network, while using a serial converter the speed is only 38.4 kbps. Using this converter is an alternative to CP5611 or similar boards.

Another similar alternative is the NETLink PRO Eth converter cable provided by Softing, which converts from **ISOTCP** to **MPI**.

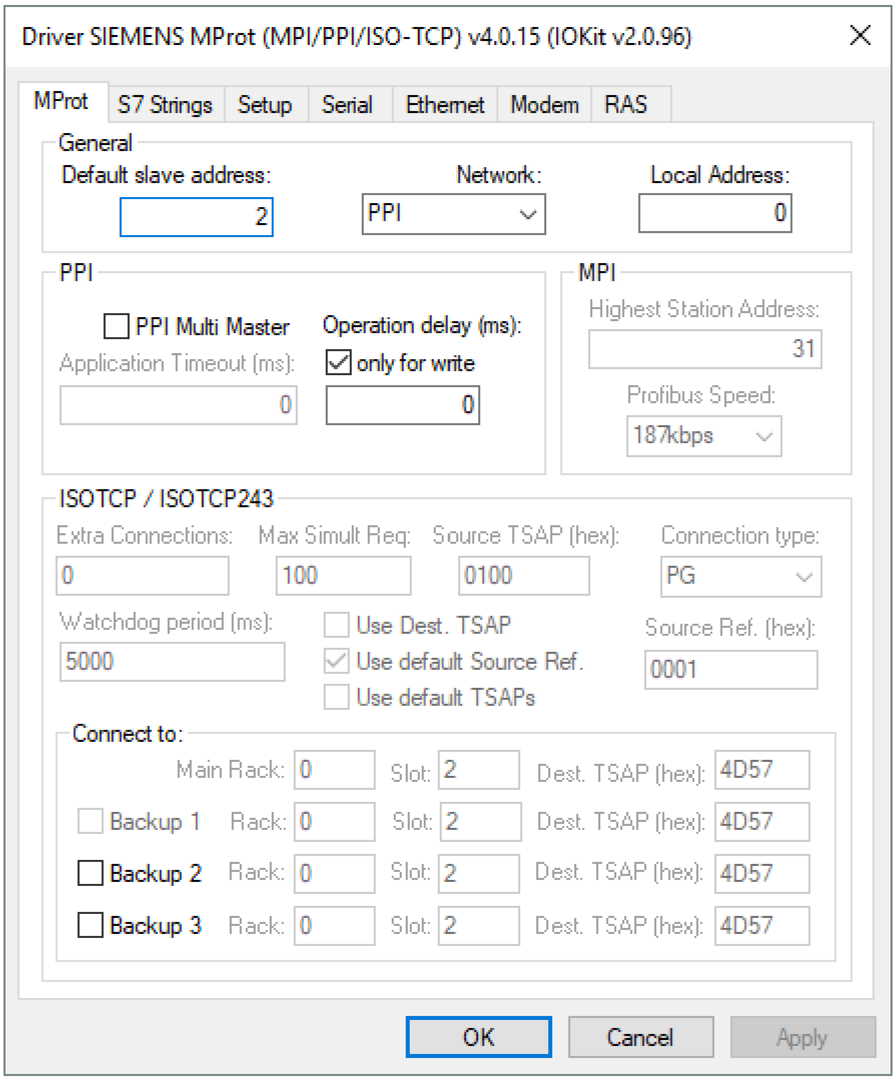
This Driver does not support using Siemens PPI/MPI adapters via USB interface.

This Driver does not support CP5611 or similar interfaces to access an MPI network. Use the S7Functions or Siemens SIMATIC.NET Drivers to communicate with these boards, by using the included OPC Server.

NOTE
M-Prot is a name created by Elipse Software to specify a Driver that supports multiple protocols. There is no relationship whatsoever with device names, protocols, or standards defined by the aforementioned manufacturers.

Driver Settings

This Driver's [P] parameters for configuration are not used. All configurations are performed on Driver's configuration window, shown on the next figure.


Driver SIEMENS MProt (MPI/PPI/ISO-TCP) v4.0.15 (IOKit v2.0.96)
MProt
S7 Strings
Setup
Serial
Ethernet
Modem
RAS
OK
Cancel
Apply

General
Default slave address: 2
Network: PPI
Local Address: 0

PPI
☐ PPI Multi Master
Application Timeout (ms): 0
Operation delay (ms): ☑ only for write
0

MPI
Highest Station Address: 31
Profibus Speed: 187kbps

ISOTCP / ISOTCP243
Extra Connections: 0
Max Simult Req: 100
Source TSAP (hex): 0100
Connection type: PG
Watchdog period (ms): 5000
☐ Use Dest. TSAP
☑ Use default Source Ref.
☐ Use default TSAPs
Source Ref. (hex): 0001

Connect to:

	Rack	Slot	Dest. TSAP (hex)
Main Rack:	0	2	4D57
☐ Backup 1	0	2	4D57
☐ Backup 2	0	2	4D57
☐ Backup 3	0	2	4D57

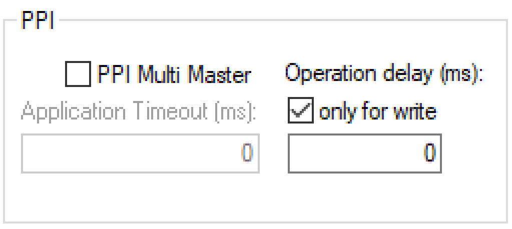
MProt tab

The available options for the **General** group are described on the next table.

Available options on the General group

OPTION	DESCRIPTION
Default Slave Address	This can be used as the default address for any Tag. Leave the *N1* parameter in 0 (zero) so that it is replaced by the default address
Network	Protocol selection: **PPI**, **MPI**, **ISOTCP**, **ISOTCP for CPU 243**, or **MPI for IBHLink converter**. The **PPI**, **MPI**, or **ISOTCP/ISOTCP243** groups on this tab are enabled or disabled according to the selected protocol
Local Address	Driver's address on the network. It can be selected arbitrarily

The available options for the **PPI** group are described on the next table.



PPI Multi Master ☐
Application Timeout (ms): 0
Operation delay (ms): 0
only for write ✓

PPI group

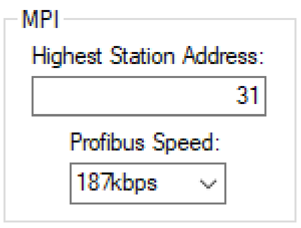
Available options on the PPI group

OPTION	DESCRIPTION
PPI Multi Master	Informs this Driver that there are other Masters on the network
Application Timeout (ms)	Maximum communication time for each Tag, in milliseconds. Available only when it is a multi-master
Operation delay (ms)	Stop time for an interval between communication operations, in milliseconds. Select the **only for write** option to apply this interval only for writing operations (please check the next note)

NOTE

The **Operation delay (ms)** option adds a minimum waiting time that must occur between the ending of a reading or writing operation and the beginning of the next one. Use a value different from 0 (zero) on this configuration only if facing communication failures caused by PLC's processing inertia. Writing operations are the most affected ones, because they are usually random, and for these operations there is the **only for write** option. If this option is not selected, the waiting time applies to reading and writing operations. If it is selected, it only applies to writing operations (recommended). Notice that adding a waiting time may slow down application's performance.

The available options for the **MPI** group are described on the next table.



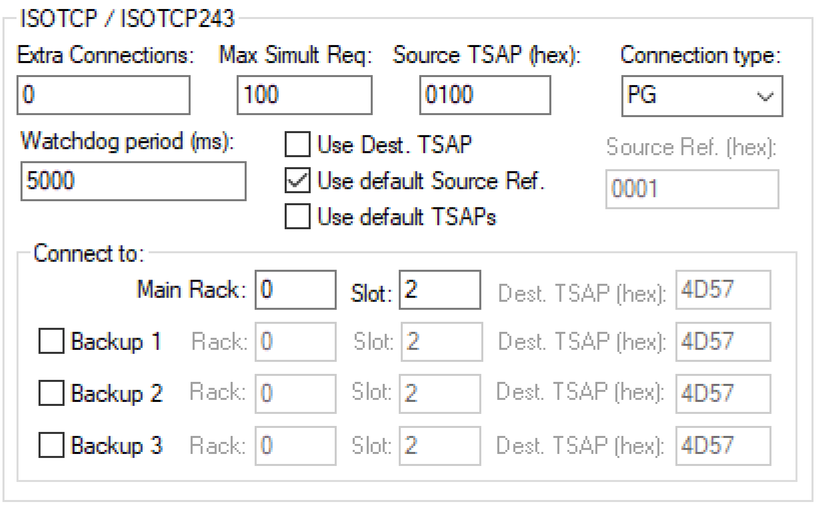
Highest Station Address: 31
Profibus Speed: 187kbps

MPI group

Available options on the MPI group

OPTION	DESCRIPTION
Highest Station Address	Indicates the greatest available address on the network, so that in **PPI** and **MPI** modes this Driver discovers other possible Masters on the network. Only the **15**, **31**, or **63** options must be added
Profibus Speed	Nominal speed of a Profibus network

The available options for the **ISOTCP / ISOTCP243** group are described on the next table.


ISOTCP / ISOTCP243

Extra Connections: 0
Max Simult Req: 100
Source TSAP (hex): 0100
Connection type: PG
Watchdog period (ms): 5000
Use Dest. TSAP [ ]
Use default Source Ref. [✓]
Use default TSAPs [ ]
Source Ref. (hex): 0001

Connect to:

	Rack	Slot	Dest. TSAP (hex)
Main Rack	0	2	4D57
[ ] Backup 1	0	2	4D57
[ ] Backup 2	0	2	4D57
[ ] Backup 3	0	2	4D57

ISOTCP / ISOTCP243 group

Available options on the ISOTCP / ISOTCP243 group

OPTION	DESCRIPTION
Extra Connections	Number of additional TCP connections that can be created to improve communication performance
Max Simult Req	Maximum number of simultaneous variables on a single request. This configuration can be used to significantly improve communication performance, as long as users use a value greater than 0 (zero) in the **Extra Connections** option
Source TSAP (hex)	A number formed by a **Word** in hexadecimal identifying the protocol's local TSAP
Connection type	Connection type: **PG**, **OP**, or **PC**. It must be selected according to CPU configuration
Watchdog period (ms)	Interval in milliseconds for **Watchdog Tag**'s expiration. If this Tag exists and its value does not vary inside this interval, then the Driver performs an IP address switch
Use Dest. TSAP	If this option is selected, enables typing values for **Dest. TSAP**. If this option is not selected, enables typing values for **Rack**, **Slot** and **Connection type**
Use default TSAPs	If this option is selected, uses default TSAP values and so disables typing configuration values of **Source TSAP**, **Dest. TSAP**, **Connection Type**, **Rack**, and **Slot**
Source Ref. (hex)	A number formed by a **Word** in hexadecimal identifying the protocol's source reference. It is only enabled when the **Use default Source Ref** option is not selected
Rack	Destination CPU's rack number
Slot	Destination CPU's slot number
Dest. TSAP (hex)	A number formed by a **Word** in hexadecimal identifying the protocol's destination TSAP
Backup 1 / Backup 2 / Backup 3	Enables typing rack and slot values for the respective backup(s) CPU(s), for use in redundancy systems with values different from the main CPU

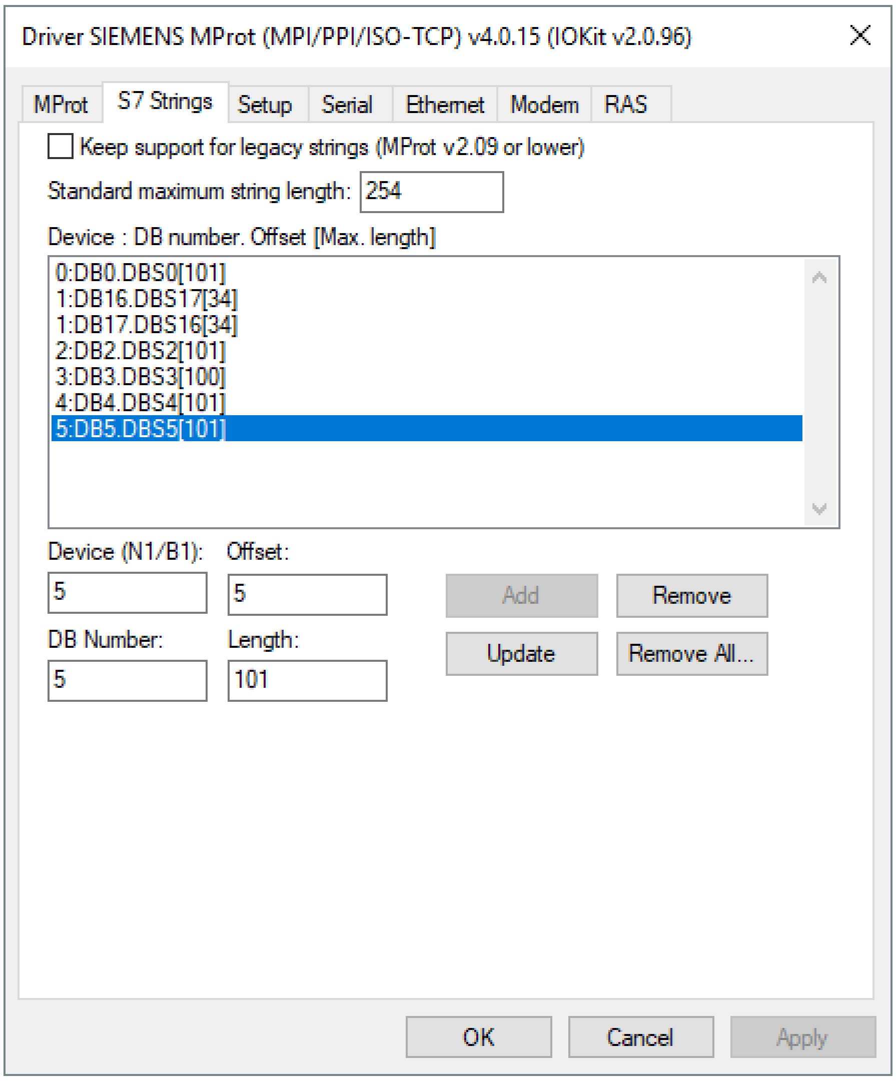
For this Driver's communication to work with the Siemens S7-1200 PLC series, users must select the **ISOTCP** option, deselect the **Use default TSAPs** option, configure the **Source TSAP (hex)** property to "0100", and define the **Connection type** option as "PG", **Rack** with 0 (zero), and **Slot** with 1 (one).

NOTES

- When selecting the **ISOTCP** or **ISOTCP243** protocols, all Tags in the Driver object must have the *N1* (or *B1*) parameter in 0 (zero) and the **Default Slave Address** parameter also in 0 (zero).
- The **Source Ref** and **Source TSAP** parameters must be considered only in very specific cases. Due to successful executions in a wide range of topologies, it is strongly recommended to keep the **Use default Source Ref** always selected and the **Source TSAP** value always as "0100".
- When the **Use Default TSAPs** option is selected together with the **ISOTCP** protocol, the **Source TSAP** value is "0100" and the **Dest. TSAP** value is "0202".
- TSAP stands for *Transport Service Access Point*, which is a terminology used by the ISO protocol.
- When using PC - PPI/MPI serial adapters, it is very common the need to configure the handshaking on the **Serial** tab of Driver's configuration window. Only the **RTS** control must be configured to **ON**. If there is any unsuccessful communication during this Driver's initial tests, it is advisable to try that change (**RTS Control** configured to **ON**) and execute the test again.

String Configuration Parameters

This tab is useful only if users must declare **Strings** with a defined maximum length, individually or generically.


Driver SIEMENS MProt (MPI/PPI/ISO-TCP) v4.0.15 (IOKit v2.0.96)

Keep support for legacy strings (MProt v2.09 or lower): unchecked

Standard maximum string length: 254

Device : DB number. Offset [Max. length]

```
0:DB0.DBS0[101]
1:DB16.DBS17[34]
1:DB17.DBS16[34]
2:DB2.DBS2[101]
3:DB3.DBS3[100]
4:DB4.DBS4[101]
5:DB5.DBS5[101]
```

Device (N1/B1): 5

Offset: 5

DB Number: 5

Length: 101

Aba S7 Strings

The available options on the **S7 Strings** tab are described on the next table.

Available options on the S7 Strings tab

OPTION	DESCRIPTION
Keep support for legacy strings (MProt v2.09 or lower)	Keeps support for old **Strings**, prior to version 2.10. By selecting this option, the old **String** format implemented on prior versions is kept, avoiding problems when updating Driver's version. It is advisable to select this option only when migrating a project whose Driver's version is 2.09 or earlier. If the project uses **Strings** after performing a version update, **String**-type Tags return reading errors from the PLC. The legacy **String** format contains a 32-byte reserved space starting from the configured offset. When working with a new project, leave this option deselected

OPTION	DESCRIPTION
Standard maximum string length	Standard maximum length of **Strings**. Fill it in with a default value configured in the PLC memory for **Strings** without a declared maximum length. For example, in S7-200 PLCs this value is equal to "254". This means that requests for **Strings** with undeclared lengths contain and indicate a fixed length of 254 characters

List of Strings' Maximum Lengths

This tab also displays a selectable list with declared **Strings** with pre-determined lengths. This list appears empty if there are no configured **Strings**. These **Strings** can be declared in the PLC memory in two ways:

- Without specifying a maximum length on declaration. Example:

```
STRING var;
```

The **String** is allocated automatically with PLC's standard maximum length.

- By specifying a maximum length on declaration. Example:

```
STRING var[50];
```

On the previous example, the **String** is allocated with a maximum length of "50". Due to that second form, this list of **String** lengths is so important.

To determine the length of a new declared **String**, users must fill in all fields correctly, as described on the next table.

Available options to configure Strings' maximum length

FIELD	DESCRIPTION
Device	PLC address. Fill it in with the same value of Tag's *N1/B1* parameter (please check topic **Standard Addressing**)
DB Number	Type the value of the DB number where the **String** is located
Offset	Type the value of the DB offset where the **String** is located
Length	Type the maximum length value of the **String**, as declared in the PLC programming

If there is already a **String** declared on the list with the same value for **Device**, **DB Number**, and **Offset**, that one is automatically selected on the table and its values are loaded to all edit fields. Three options help users when editing **String** data on the list:

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

Click **OK** to confirm all configurations listed and close this window. Click **Remove All** to remove all data from this list.

NOTE

When declaring Tags with Symbolic Addressing parameters, there is no need to fill in this list with **String** declarations. Their length can be specified on the symbol parameter available in the Tag.

Tag Reference

This section contains information about the configuration of Tags by **Symbolic Addressing** and by **Standard Addressing** (*N/B* parameters). It also contains references to **Interface Tags on Extra ISOTCP Connections**.

Configuration by Syntactical Parameters

Use the following syntax for each field in **E3** or **Elipse Power**:

- **Device**: Insert the device's address on the network. If it is equal to 0 (zero) and the selected protocol is different from **ISOTCP** or **ISOTCP243**, then it is replaced by the **Default Slave Address**. If the selected protocol is **ISOTCP** or **ISOTCP243**, this value must be left as 0 (zero). The **Device** field may also be left blank, as long as it is inserted in the **Item** field before the colon symbol
- **Item**: This field must obey one of the defined syntaxes described next

Use the following general syntax if area is not equal to **DB**. Values inside brackets are optional:

```
<[Device:]><Area><[Type]><Address>[.Bit]
```

Where:

- **Device**: PLC address as exposed in the **Device** item, if it was not informed in that field
- **Area**: Data area inside the PLC. The following options can be used:
  - **S**
  - **SM**
  - **AI** (*Analog Input*)
  - **AQ** (*Analog Output*)
  - **C** (*Counter*)
  - **T** (*Timer*)
  - **I** (*Digital Input*)
  - **Q** (*Digital Output*)
  - **M** (*Memory*)
  - **HC** (*High Speed Counter*)
- **Type**: Data type to read. The next table shows all possible symbols for these data types

Available options for data types

DATA TYPE	MEANING
X	Used to extract a bit from a byte
B	**Byte**
W	**Word**
D	**DWord**
I	**Int**
LI	**DInt**
F	**Float**
S	**String**
S5T	**S5Time**

- **Address**: Numerical address to read
- **Bit**: Optional informing the bit from a word to read or write (between zero and 31)

Example:

```
(PLC 4, bit 1 from memory at address 10)
Device: Blank - Item 4:M10.1
```

If the data area is equal to **DB** (also known as **V**), use the following syntax. Values inside brackets are optional:

```
<[Device:]>DB<DBNumber>:<Type><Address><[.Bit]>
```

- **Device**: Refers to the same optional item of the general syntax
- **DBNumber**: Fill it in with the DB number. If the memory contains a single or unspecified DB block, fill it in with value 1 (one)
- **Address**: Numerical address (offset) to read
- **Bit**: Optional value informing the bit of a type to read or write (between zero and 31)

Available options for data types on DB data areas

DATA TYPE	MEANING
DBX	Used to extract a bit from a byte in a **DB**
DBB	Used to read or write a byte in a **DB**
DBW or DW	Used to read or write a **Word** in a **DB**
DBD or DD	Used to read or write a **Double Word** in a **DB**
DBI or DI	Used to read or write a **Int** in a **DB**
DBLI or DLI	Used to read or write a **DInt** in a **DB**
DBF or DF	Used to read or write a **Floating Point** (32-bit real) in a **DB**
DBS or DS	Used to access a **String** in a **DB**
DBS5T	Used to access an **S5Time**-type timer in a **DB**

Examples:

```
(PLC 2, Word starting at address 20 of DB1)
Device: 2 - Item: DB1:DW20
(Same as the previous one, but Device was informed in the Item field)
Device: Empty - Item: 2:DB1:DW20
(PLC 7, DB 5, bit 2 from byte 7)
Device: Empty - Item: 7:DB5:DBX7.2
```

The syntax for **String** data types in the **DB** data area is the following:

```
<[Device:]>DB<DBNumber>:DBS<Address><[Maximum length]>
```

Where:

- **Device, DBNumber, and Address**: Refer to the same items of the general syntax

- **Maximum length**: Optional informing the maximum length declared on the **String**. If it is not informed, then it considers the maximum default length of the **String** as informed on the **Strings** configuration window

Examples of syntax for **Strings**:

```
(PLC 2, String starting at address 16 of DB17,
using PLC's maximum default length)
Device: 2 - Item: DB17:DBS16
(same as the previous one, but Device was informed in the Item field
and with a maximum allocated length of 25 characters)
Device: Empty - Item: 2:DB17:DBS16[25]
(PLC 4, String starting at address 100 of DB10,
with a maximum allocated length of 50 characters)
Device: Empty - Item 4:DB10:DS100[50]
```

Configuration by Numerical Parameters (N/B)

Use the default syntax described on the next table for all Tags and Blocks.

Default syntax for Tags and Blocks

PARAMETER	DESCRIPTION
N1/B1	PLC address. If it is equal to 0 (zero) and the selected protocol is different from **ISOTCP** or **ISOTCP243**, then it is replaced by the **Default Slave Address**. If the selected protocol is **ISOTCP** or **ISOTCP243**, this value must be left as 0 (zero)
N2/B2	Data type and Data area (please check the next tables). This value must be composed by the data type multiplied by 100 plus the data area (its formula is *N2/B2 = DataType × 100 + Area*)
N3/B3	If the selected area is **V** (**DB**), fill it in with the number of the DB block. Otherwise, leave it in 0 (zero). If the memory contains a single or unspecified DB block, fill it in with the value 1 (one)
N4/B4	DB block's address in the data area or offset. To use data types that require more than one byte, use addresses that are multiples of two for two-byte types (signed or unsigned 16-bit) and multiples of four for four-byte types (signed or unsigned 32-bit and 32-bit floating point)

Available options for data types

DATA TYPE	MEANING
0	Data area's default
1	**BOOL** (Boolean)
2	**BYTE** (unsigned 8-bit)
3	**WORD** (unsigned 16-bit)
4	**INT** (signed 16-bit)
5	**DWORD** (unsigned 32-bit)
6	**DINT** (signed 32-bit)

DATA TYPE	MEANING
7	**REAL** (32-bit floating point - IEEE 754)
8	**STRING** (please check the **note** later on this topic)
12	**S5TIME** (time in seconds, 32-bit floating point - IEEE 754, please check the **note** later on this topic)

Available options for data areas

DATA AREA	MEANING
0	**S**
1	**SM**
2	**AI** (Analog Input)
3	**AQ** (Analog Output)
4	**C** (Counter)
5	**T** (Timer)
6	**I** (Digital Input)
7	**Q** (Digital Output)
8	**M** (Memory)
9	**V** (DB)
10	**HC** (High Speed Counter)

NOTES

- For **S5Time** data types, the value to be filled in is always in seconds, as a 32-bit floating point. The range of values different from zero is between 0.01 and 9990.0 seconds. The time base is filled in or interpreted automatically.
- In the **PPI** protocol there is a limitation in the I/O Block for data in bytes. For reading, the maximum allowed is 224 bytes, and for writing it is 218 bytes. This means, respectively, that for **Word** data types (16 bits), a Block cannot have more than 112 and 109 Elements. For **DWord** data types (32 bits), a Block cannot have more than 56 and 54 Elements, and so on.
- If **Rack** and **Slot** definitions are unknown for Tag addressing in the **ISOTCP** protocol, please check the article *KB-39019: Rack and Slot settings*, on **Elipse Knowledgebase**.

Interface Tags on Extra ISOTCP Connections

By opting to use extra ISOTCP connections with the **Extra Connections** parameter on the **Driver's configuration window**, these connections can be controlled and monitored by three Interface-specific Tags: **Physical Layer Status**, **IPSelect**, and **IPSwitch**.

NOTE

These Tags cannot be used when the **Extra Connections** parameter is equal to 0 (zero). In this case, use the corresponding **IOKit** Tags, with the same name, whose usage can be checked on topic **Documentation of I/O Interfaces**.

Physical Layer Status (MProt)

Read-Only

Configuration by numerical parameters

PARAMETER	VALUE
N1	-2
N2	0 (zero)
N3	0 (zero)
N4	2

Configuration by syntactical parameters

PARAMETER	VALUE
Item	MProt.IO.PhysicalLayerStatus

This Tag indicates the status of the connection on the physical layer. Its possible values are the following:

- **0**: Physical layer disconnected
- **1**: Physical layer connected

IPSelect (MProt)

Reading and Writing

Configuration by numerical parameters

PARAMETER	VALUE
N1	-2
N2	0 (zero)
N3	4
N4	0

Configuration by syntactical parameters

PARAMETER	VALUE
Item	MProt.IO.Ethernet.IPSelect

Indicates the active IP address. Its possible values are the following:

- **0**: The main IP address is selected (active)
- **1**: The alternative IP address (backup) is selected (active)

If the Ethernet interface is connected, this Tag indicates which one of the two configured IP addresses is in use. If the interface is disconnected, this Tag indicates which IP address is used first in the next attempt to connect.

During the connection process, if the active IP address is not available, then **IOKit** tries to connect to the other IP address. If the connection to the alternative IP address succeeds, then this IP address is set as the active one (automatic switchover).

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

IPSwitch (MProt)

Write-Only

Configuration by numerical parameters

PARAMETER	VALUE
N1	-2
N2	0 (zero)
N3	4
N4	1

Configuration by syntactical parameters

PARAMETER	VALUE
Item	MProt.IO.Ethernet.IPSwitch

Writing any value to this Tag forces a manual switchover. If the main IP address is active, then the backup IP address is activated, and vice versa. This forces a reconnection to the specified IP address if the Driver is currently connected. If the Driver is disconnected, that configures the active IP address for the next attempt to connect.

Watchdog Tag

This feature is useful only for redundancy architectures that rely on a variable in the CLP to be elected to indicate the time to switch the destination IP address. In the vast majority of cases, this type of Tag is not needed. If this is the case to apply to a project, this Tag must be a **Boolean** variable (**BIT**) that must vary its value before the configured time expires, and logically must be read with a fairly scan time lower than the expiration time (the **Watchdog period (ms)** option on the **ISOTCP / ISOTCP243** group of the **MProt** tab on the configuration window). Whenever this variable does not vary by that period, the IP address switch is then executed. To configure a Tag as a **Watchdog**, add a "/watchdog" **String** at the end of the syntax parameter **Item**, such as "DB2:DBX0.0/watchdog".

SOE Collecting

This section contains specific information about SOE's event collecting.

Preparing for SOE Collecting

Before using SOE Collecting Tags, users must prepare the PLC by creating a DB Table (**V** area) and develop a programmable logic compatible with all SOE collecting procedures developed for this Driver.

Table of SOE Events

This table aims to configure the size of the event buffer and manage their input and output in a circular buffer routine. This table is constantly updated by both the PLC and the Siemens MProt Driver.

The Table of SOE Events must contain registers on control and storage of events, based on the data structure described on the next table.

Data structure

ADDRESS	DESCRIPTION	DATA TYPE
0.0		**STRUCT**
+0.0	Table status	**WORD** (unsigned 16-bit)
+2.0	Recording pointer	**WORD** (unsigned 16-bit)
+4.0	Acquisition status	**WORD** (unsigned 16-bit)
+6.0	Maximum limit of items on the circular buffer	**WORD** (unsigned 16-bit)
+8.0	Circular buffer	**ARRAY[1..n]** (limit of user-defined items)
+0.0		**STRUCT**
+0.0	**TIMESTAMP_LOLO** (year)	**WORD** (unsigned 16-bit)
+2.0	**TIMESTAMP_LOHI** (day and month)	**WORD** (unsigned 16-bit)
+4.0	**TIMESTAMP_HILO** (hour and minute)	**WORD** (unsigned 16-bit)
+6.0	**TIMESTAMP_HIHI** (second and millisecond)	**WORD** (unsigned 16-bit)
+8.0	Value of event type 1	Event's data type (user-defined)
+n.0	Value of event type 2	Repeats the same data type
+n.0	Value of event type 3	Repeats the same data type
+n.0	Value of event type n	Repeats the same data type
=n.0		**END_STRUCT**
=n.0		**END_STRUCT**

Description of Event Control Registers

- **Table Status**: It must be kept exclusively by the PLC, indicating the number of events available for reading in the circular buffer. It must be updated by the PLC whenever new events are added to the circular buffer, or after completing the collecting of events by the application, which can be detected when **Acquisition Status** changes
- **Recording Pointer**: It must be kept exclusively by the PLC, indicating the index, starting at zero, of the position where the next event must be inserted. The index must be incremented by the PLC whenever a new event is inserted in the circular buffer, then returning to index zero after reaching the maximum limit of the circular buffer

- **Acquisition Status**: It must be kept by the PLC and by the MProt Driver, indicating the number of records already read at every transaction. After each collecting, the MProt Driver writes to this register the number of events that it could read. When detecting this change, the PLC must immediately subtract this value written by the MProt Driver from the **Table Status** and then reset the **Acquisition Status**

- **Maximum Limit of Items of the Circular Buffer**: A constant value that specifies the maximum limit of events to store in the circular buffer before the pointer moves back to index 0 (zero). It must contain exactly the limit value of the Array resized for events of the circular buffer

Description of Event Storage Registers

- **TIMESTAMP**: Time when the event occurred

- **Event Value**: Value of the occurred event, which can be composed by one or *n* values (all with the same data type), in which they are grouped together for the same **TIMESTAMP** generated when an event occurs

TIMESTAMP format

A **TIMESTAMP** is represented by four **Words**, according to the data structure described on the next table.

Data structure

WORD	CONTENT	RANGE
0	Year	Between 0 (zero) and 65535
1	Day and month	ddddddddmmmmmmmm
2	Hour and minute	hhhhhhhhmmmmmmmm
3	Second and millisecond	ssssssmmmmmmmmmm

- The first **Word** contains an integer value referring to a year

- The second **Word** is divided into a high byte to represent a day and into a low byte to represent a month

- The third **Word** is divided into a high byte to represent hours and into a low byte to represent minutes

- The fourth **Word** uses the six highest bits to represent seconds and the 10 lowest bits to represent milliseconds

Acquisition Procedure

The PLC must start inserting events in ascending order, starting from table's base address, referring to the beginning of the circular buffer. At every new event inserted, the recording pointer must be incremented, starting to point to the next buffer's available address.

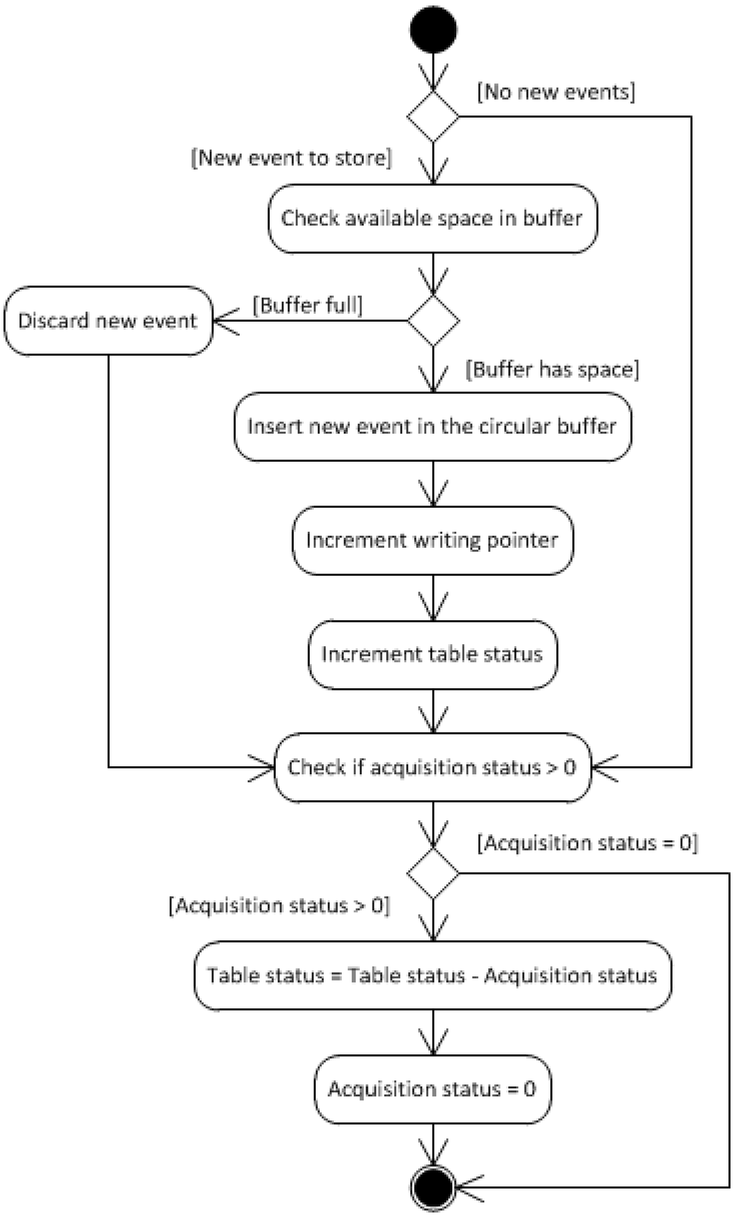
This Driver performs an event reading from the oldest to the newest. The starting address for reading is calculated by this Driver using the value of **Recording Pointer** and **Table Status**.

If the number of available events is greater than the maximum allowed in a single protocol's communication frame, this Driver performs multiple block readings, updating the value of **Acquisition Status** at the end of the process with the total amount of events read.

When detecting that this Driver wrote a value greater than 0 (zero) to **Acquisition Status**, the PLC must immediately subtract the value of **Acquisition Status** from the value of **Table Status** and then reset **Acquisition Status**.

The PLC can insert new events on the table during PLC's acquisition process, as long as there is no overflow in the circular buffer, then incrementing **Table Status**.

The next figure presents a flowchart, as a UML Activity Diagram, with a suggested implementation for this PLC logic.



SOE flowchart

SOE Collecting Tags

The SOE collecting of events is performed by using the Tags described next, by using an ISOTCP communication with the PLC.

Block Tag for Control Register (Read-Only)

- **B1**: 0 (zero)
- **B2**: 391 (Data type = 3 x 100 + Data area = 91)
- **B3**: DB block number. If memory contains a single or unspecified DB block, fill it in with value 1 (one)
- **B4**: Not used

The Block Tag to query Control Registers must contain four Elements to return the following values:

- **Element 1**: Table status
- **Element 2**: Recording pointer
- **Element 3**: Acquisition status
- **Element 4**: Maximum limit of items on the circular buffer

For a description of each one of these Control Registers, please check topic **Preparing for SOE Collecting**.

Tag Block for Data Collecting (Read-Only)

- **B1**: 0 (zero)
- **B2**: Data type x 100 + Data area = 90
- **B3**: DB block number. If memory contains a single or unspecified DB block, fill it in with value 1 (one)
- **B4**: Not used

The Block Tag for Data Collecting must contain a number of Elements corresponding to the number of values of *n*-event type that compose a single event. If this event is composed of a single value, resize the Block Tag for Data Collecting with a single Element. If this event is composed by two values, the Block Tag must be resized to two Elements, and so on. Use Block Tag's *B2* parameter to indicate a data type associated to event values.

NOTE

- All values that compose an event must have the same data type, as well as every PLC's DB table must be filled in with the same event type.
- Data Areas **90** and **91** exist only for use at the logical user application level. At the protocol level, both Tags work in PLC's Data Area **9** (**DA_V**) automatically.

Documentation of I/O Interfaces

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

Driver Configuration

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

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

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

1. Open the Organizer.
2. Select the Driver on Organizer's tree.
3. Click **Extras** on **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 Driver for each serial port.

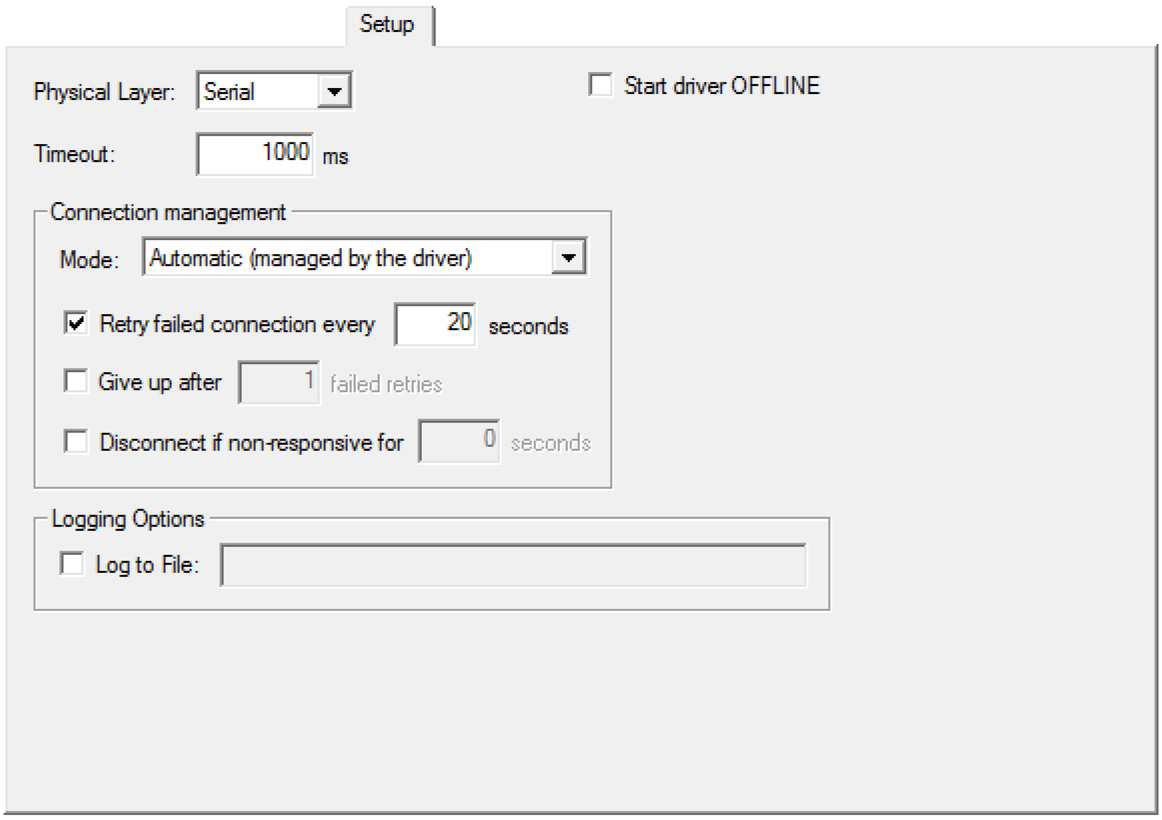
Configuration Dialog Box

The I/O Interfaces dialog box 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 each Driver) on the configuration dialog box.

Setup Tab

The **Setup** tab contains Driver's general configurations. This tab is divided into three distinct parts:

- **General configurations**: Configurations of 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



Physical Layer: Serial

Start driver OFFLINE [ ]

Timeout: 1000 ms

Connection management

Mode: Automatic (managed by the driver)

[✓] Retry failed connection every 20 seconds

[ ] Give up after 1 failed retries

[ ] Disconnect if non-responsive for 0 seconds

Logging Options

[ ] Log to File:

Setup tab

General options on Setup tab

OPTION	DESCRIPTION
Physical Layer	Select the physical layer on the 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 a byte (any byte from reception's buffer).
Start driver OFFLINE	Select this option so that the Driver starts in **Offline** mode (stopped). This means that the I/O interface is not created until this Driver is configured to **Online** mode (using a Tag in an application). This mode enables a dynamic configuration of an I/O interface at run time. Please check topic **Working Offline** for more details.

Options on Connection management group

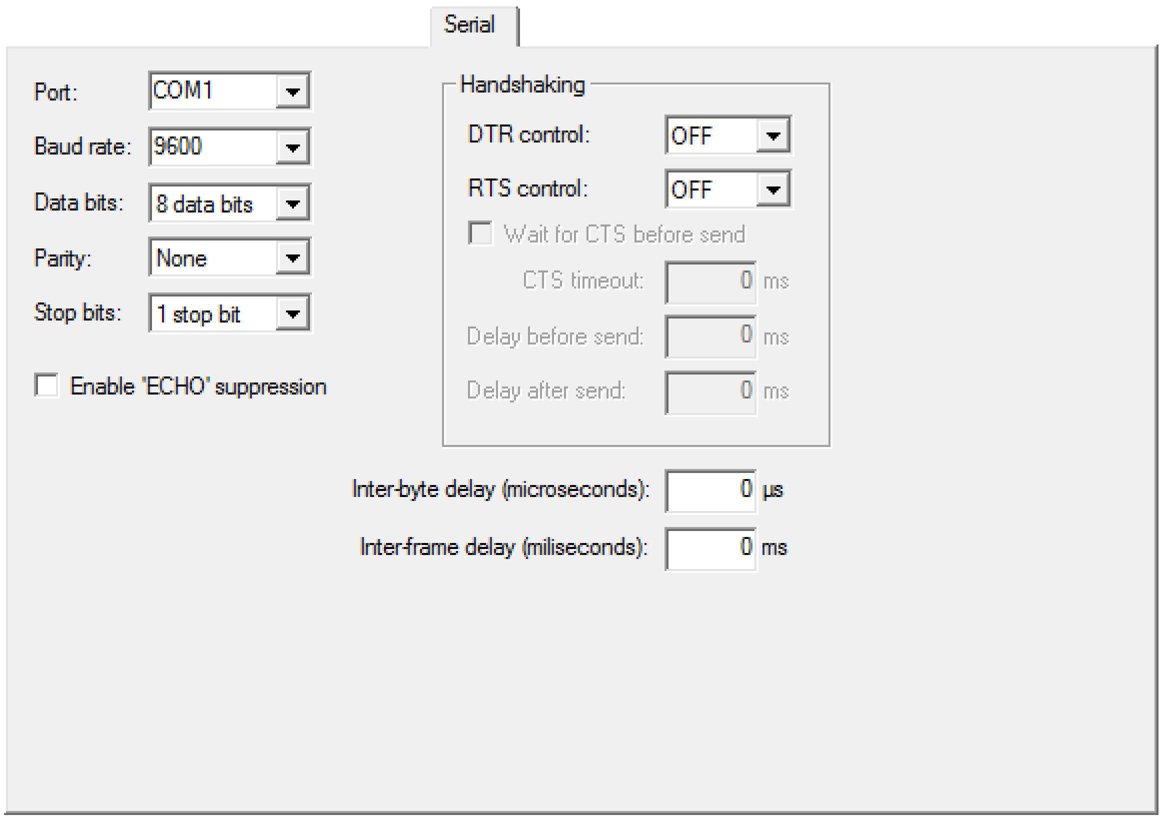
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage the connection. Please check topic **Driver Statuses** for more details.
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, the Driver keeps retrying until the 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, the 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 Logging Options group

OPTION	DESCRIPTION
Log to File	Enable this option and configure the name of the file to write the log. Log files can be large, so use this option for short periods of time, only for test and debugging purposes. If the **%PROCESS%** macro is used in the log file name, it is replaced by the ID of the current process. This option is particularly useful when using several instances of the same Driver in **E3**, thus allowing each instance to generate a separate log file. For example, when configuring this option as **c:\e3logs\drivers\sim_%PROCESS%.log**, a file **c:\e3logs\drivers\sim_00000FDA.log** is generated 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 as **c:\e3logs\drivers\sim_%DATE%.log**, a file **c:\e3logs\drivers\sim_2005_12_31.log** is generated in 12/31/2005 and a file **c:\e3logs\drivers\sim_2006_01_01.log** is generated in 01/01/2006.

Serial Tab

Use this tab to configure parameters of the **Serial** Interface.


Serial

Port: COM1
Baud rate: 9600
Data bits: 8 data bits
Parity: None
Stop bits: 1 stop bit
☐ Enable 'ECHO' suppression

Handshaking
DTR control: OFF
RTS control: OFF
☐ Wait for CTS before send
CTS timeout: 0 ms
Delay before send: 0 ms
Delay after send: 0 ms

Inter-byte delay (microseconds): 0 µs
Inter-frame delay (miliseconds): 0 ms

Serial tab

General options on Serial tab

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

OPTION	DESCRIPTION
Inter-frame delay (milliseconds)	Defines a delay between packets sent or received by the I/O Interface, in thousandths of a second (1000 is equal to a second). This delay is applied if the I/O Interface sends two consecutive packets, or between a received packet and the next sending.

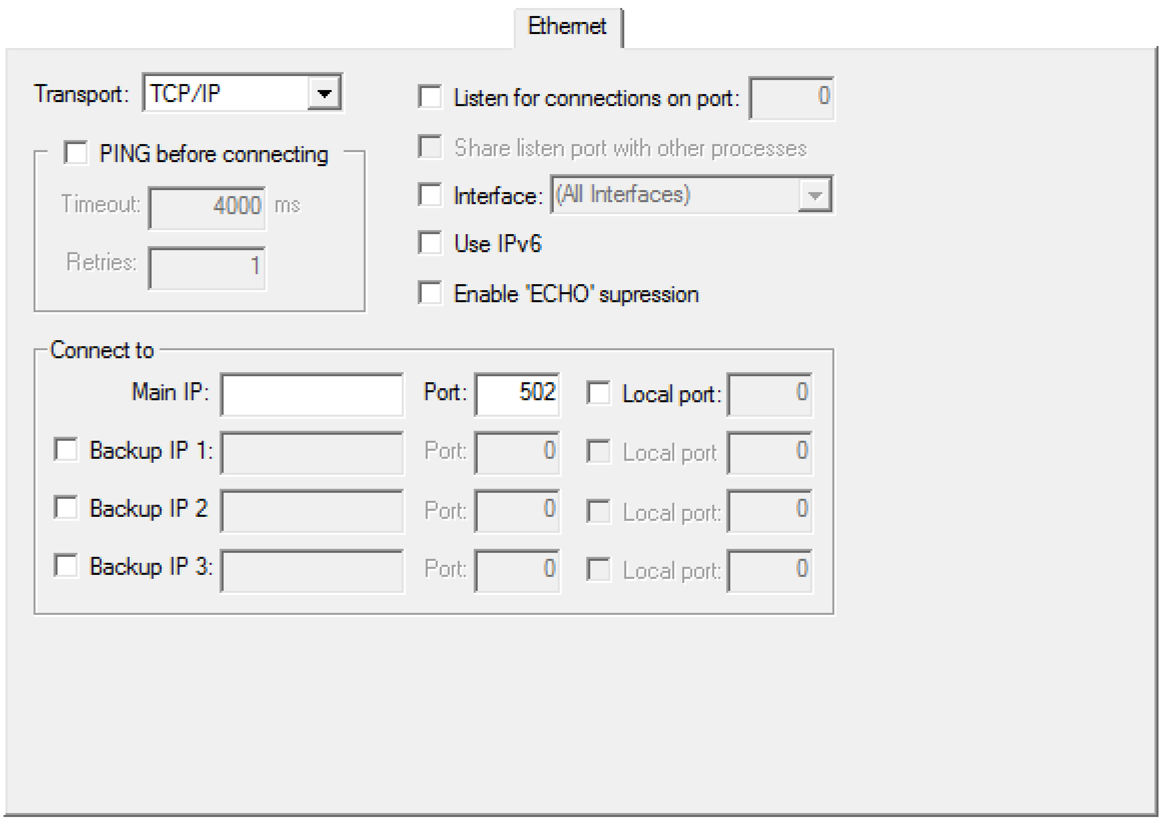
The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process (controlling when data can be sent or received via serial line). Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works with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

Available options on Handshaking group

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

Ethernet Tab

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



Ethernet tab

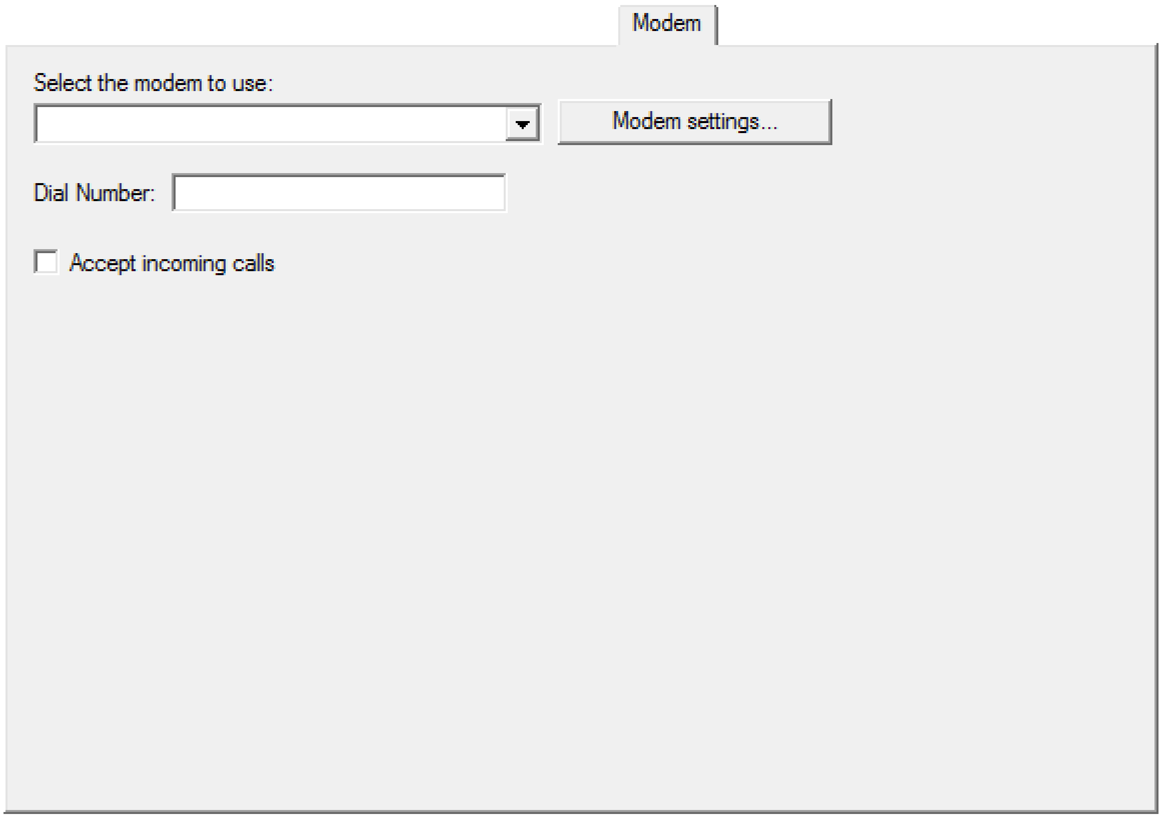
Available options on Ethernet tab

OPTION	DESCRIPTION
Transport	Select **TCP/IP** for a TCP socket (stream). Select **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, the 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 listen port with other Drivers and processes
Interface	Select the local network interface (identified by its IP address) that is used by the Driver to establish and receive connections, or select the **(All Interfaces)** item to use any local network interface
Use IPv6	Check this option to force the Driver to use IPv6 addresses on all Ethernet connections. If this option is unchecked the Driver will work with IPv4 addresses
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 details
Main IP **Backup IP 1** **Backup IP 2** **Backup IP 3**	These options allow configuring up to four IP addresses for a remote device: • **IP**: Type an IP address for the remote device. This can be an IP address separated by dots, as well as a URL. For a URL, the Driver uses the available DNS service to map that URL to an IP address. For example, "192.168.0.13" or "Server1" • **Port**: Type an IP port for a remote device (from 0 to 65535) • **Local port**: Select this option to use a fixed local port when connecting to a remote device
PING before connecting	Enable this option to execute a **ping** command (check if 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): • **Timeout**: Specify the number of milliseconds to wait for a reply from the **ping** command. Users must use the **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 (between one and 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

Modem Tab

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



Modem tab

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

Available options on Modem tab

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

RAS Tab

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

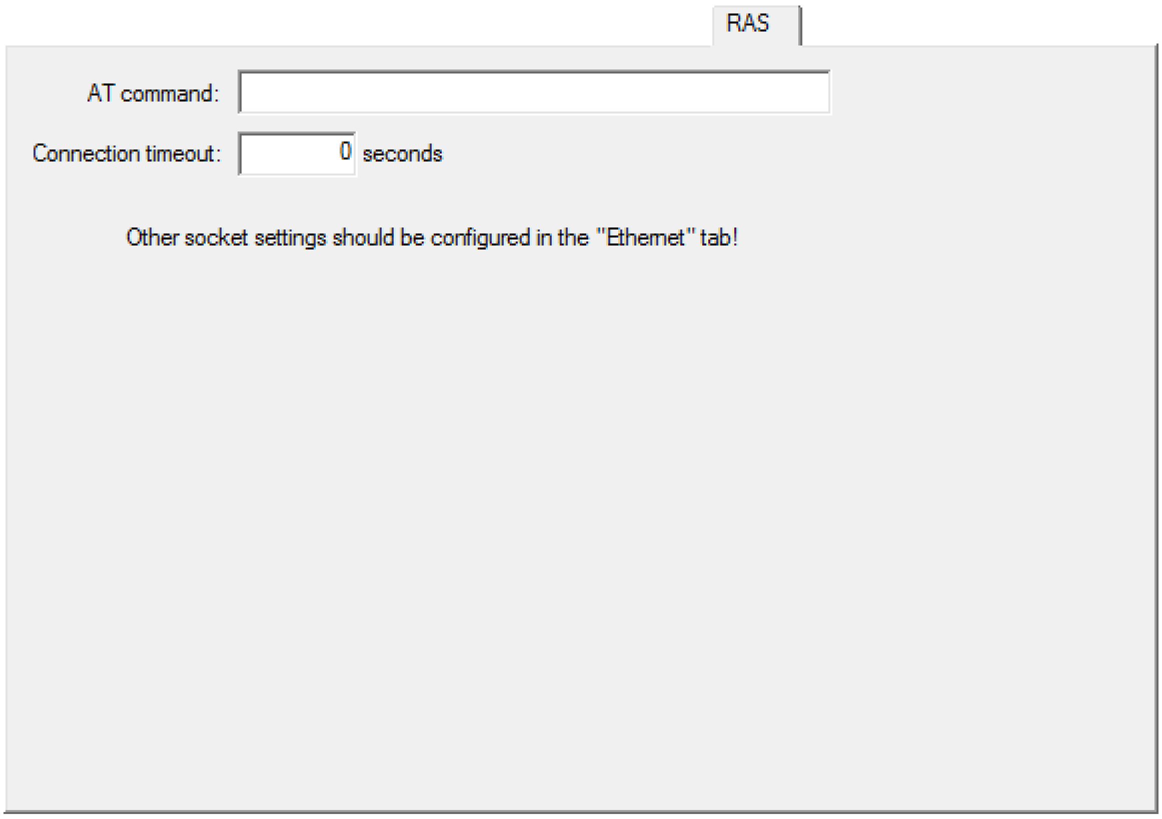
The **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

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

1. Clear the socket (remove any TELNET greeting message received from the RAS device).
2. Send an **AT** dial message (in ASCII) in the socket.
3. Wait for a **CONNECT** reply.
4. If the time-out expires, the connection is aborted.
5. If the **CONNECT** reply is received within the time-out, the socket is available for communication with the device (connection was established).

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

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



AT command:

Connection timeout: 0 seconds

Other socket settings should be configured in the "Ethernet" tab!

RAS tab

Available options on RAS tab

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

General Configurations

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

I/O Tags

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

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

IO.IOKitEvent

Type of Tag	Block Tag
Type of Access	Read-Only
B1 Parameter	-1
B2 Parameter	0
B3 Parameter	0
B4 Parameter	1
Size Property	4
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
  - **0**: Information
  - **1**: Warning
  - **2**: Error
- **Element 1**: Source of event
  - **0**: Driver (specific of a Driver)
  - **-1**: IOKit (generic events of I/O Interfaces)
  - **-2**: **Serial** Interface
  - **-3**: **Modem** Interface
  - **-4**: **Ethernet** Interface

    - **-5**: **RAS** Interface

- **Element 2**: Error number (specific for each source of event)
- **Element 3**: Event message (**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
N2 Parameter	0
N3 Parameter	0
N4 Parameter	2
String Configuration	IO.PhysicalLayerStatus

This Tag indicates the status of the physical layer. Its possible values are the following:

- **0**: Physical layer stopped (the 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 (the Driver is in **Online** mode, but the physical layer is not connected. If the **Connection management** option is configured as **Automatic**, the physical layer can be connecting, disconnecting, or waiting for a reconnection attempt. If the **Connection management** option is configured as **Manual**, then the physical layer remains in this status until forced to connect)
- **2**: Physical layer connected (the physical layer is ready for use). This **DOES NOT** mean the device is connected, only the access layer is working

IO.SetConfigurationParameters

Type of Tag	Block Tag
Type of Access	Read-Only
B1 Parameter	-1
B2 Parameter	0
B3 Parameter	0
B4 Parameter	3
Size Property	2
ParamItem Property	IO.SetConfigurationParameters

Use this Tag to change any property of Driver's configuration dialog box at run time (the complete list of properties can be found on the specific topic of each Interface).

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 Driver's configuration dialog box. Users can write to a PLC Tag or to a Block Tag containing the parameters to change (writings of individual Block Elements are 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 three parameters, then the size of the Block must be 6 (3 * 2). The first Element is the property's name (as a **String**) and the second Element is the property's value. Check this script in **Elipse SCADA**:

```
// 'Block' must be a Block Tag with automatic reading,
// scan reading, and automatic writing 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 **E3**, the ability to create arrays at run time allows using an I/O Tag as well as a Block Tag. Users can use Driver's **Write** method to send all parameters to the Driver, without creating a Tag. Check these examples:

```
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 elements of the array are ignored by the 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 Driver's log or use the *writeStatus* parameter of the **WriteEx** method to find out the exact cause of the 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 "Failure when configuring Driver parameters: " + strError
End If
```

IO.WorkOnline

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

This Tag informs Driver's current status and allows starting or stopping the physical layer.

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

In the next example (using **E3**), the Driver is configured to **Offline** mode, its COM port is changed, and then configured to **Online** mode again:

```
' Configure to Driver 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 Driver to Online mode
Driver.Write -1, 0, 0, 4, 1
```

The **Write** method can fail when configuring the Driver to **Online** mode (writing the value one). In this case, the 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)
- Driver may have run out of memory
- Physical layer probably did not create its working thread (search the log file for the message "Failed to create physical layer thread!")
- Physical layer could not start. The cause of failure depends on the type of physical layer. It can be an invalid serial port number, failure when starting Windows Sockets, failure when starting TAPI (modem), etc. 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 (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 the Connection:

- **0**: Automatic mode (the Driver manages the connection)
- **1**: Manual mode (the application manages the connection)

IO.GiveUpEnable

When configured to True, defines a maximum number of reconnection attempts. If all reconnection attempts fail, the Driver enters the **Offline** mode. When configured to False, the 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), the Driver tries only one reconnection when the reconnection is lost. If this one fails, the 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 inactivity. If the physical layer is inactive for this period of time, it is 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

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

- **N or None**: Does not use a physical interface (the 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. The 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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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
N2 Parameter	0
N3 Parameter	0
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 **Ethernet** Interface.

I/O Tags

Tags of Ethernet Interface (N2/B2 = 4)

The Tags described next allow controlling and identifying the **Ethernet** Interface at run time (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
N4 Parameter	0 (zero)
String Configuration	IO.Ethernet.IPSelect

Indicates the active IP address. Possible values are the following:

- **0**: The main IP address is selected

- **1**: The alternative (backup) IP address is selected
- **2**: The alternative (backup) IP address 2 is selected
- **3**: The alternative (backup) IP address 3 is selected

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

During the connection process, if the active IP address is not available, the I/O Interface tries to connect using the next alternative 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 IP address or **1, 2, 3**: Alternative IP address) if the Driver is currently connected. If the Driver is disconnected, this Tag configures the active IP address for the next connection attempt.

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
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 Driver tries to connect to each one of the alternative (backup) IP addresses and back to the main IP address until a connection is established.

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

Properties

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

NOTE

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

IO.Ethernet.AcceptConnection

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

IO.Ethernet.BackupEnable

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

IO.Ethernet.BackupLocalPort

Local port number to be used when connecting to the alternative IP address of the destination device. Used only if **IO.Ethernet.BackupLocalPortEnable** is True.

IO.Ethernet.BackupLocalPortEnable

Configure to True to force the use of a specific local port when connecting to the backup IP address of a remote device. Configure to False to let windows find dinamically an available local port.

IO.Ethernet.BackupIP

Alternative (backup) IP address of the destination 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.BackupPort

Port number of the alternative IP address of the destination device (used with the **IO.Ethernet.BackupIP** property).

IO.Ethernet.IPFilter

A List with a comma-separated IPv4 or IPv6 addresses, which defines from which addresses the 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 the IP addresses. To block an IP address or a range of IP addresses, use the tilde ("~") character at the beginning of the address. 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 range from 192.168.0.41 to 192.168.0.50
- **192.168.0.***: Accepts connections from IPv4 addresses in the range from 192.168.0.0 to 192.168.0.255
- **fe80:3bf:877::*:* (expands to fe80:03bf:0877:0000:0000:0000:*:*)**: Accepts connections from IPv6 addresses in the range from fe80:03bf:0877:0000:0000:0000:0000:0000 to 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 range from 192.168.0.0 to 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 the 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 the message "Blocked incoming socket connection from {IP}!".

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

IO.Ethernet.ListenIP

A IP address of the local network interface that the Driver uses to establish and receive connections. Leave this property empty to use any local network interface.

IO.Ethernet.ListenPort

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

IO.Ethernet.MainIP

A IP address of the destination 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 the destination 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. Configure to False to use any available local port.

IO.Ethernet.MainPort

9 Number of the IP port on the destination 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 the destination 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 the listen port with other Drivers and processes or False to open the listen port in exclusive mode. To successfully share a listen port, all Drivers and processes that use that port must open it in shared mode. When a listen 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 in 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 the following:

- **T or TCP**: Uses the TCP/IP protocol
- **U or UDP**: Uses the UDP/IP protocol

IO.Ethernet.UseIPv6

☑ Configure in True to use IPv6 addresses on all Ethernet connections. Configure in False to use IPv4 addresses (this is the default value).

Modem Interface Configuration

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

I/O Tags

Tags of Modem Interface (N2/B2 = 3)

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

IMPORTANT

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

IO.TAPI.ConnectionBaudRate

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1
N2 Parameter	0
N3 Parameter	3
N4 Parameter	5
String Configuration	IO.TAPI.ConnectionBaudRate

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

IO.TAPI.Dial

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

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

IO.TAPI.HangUp

Type of Tag	I/O Tag
Type of Access	Write-Only
N1 Parameter	-1
N2 Parameter	0
N3 Parameter	3
N4 Parameter	4
String Configuration	IO.TAPI.HangUp

Any value written to this Tag turns the current call off.

NOTE

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

IO.TAPI.IsModemConnected

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

This Tag indicates modem's connection status. Possible values are the following:

- **0**: The modem is not connected, but it may be performing or receiving an external call
- **1**: The modem is connected and the Driver completed or received an external call successfully. While it is in this status, the physical layer can send or receive data

IO.TAPI.IsModemConnecting

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1
N2 Parameter	0
N3 Parameter	3
N4 Parameter	6
String Configuration	IO.TAPI.IsModemConnecting

This Tag indicates the connection status of a modem, with more details than the **IO.TAPI.IsModemConnected** Tag. Possible values are the following:

- **0**: Modem is not connected
- **1**: Modem is connecting (performing or receiving an external call)
- **2**: Modem is connected. While in this status, the physical layer can send or receive data
- **3**: Modem is disconnecting the current call

IO.TAPI.ModemStatus

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1
N2 Parameter	0
N3 Parameter	3
N4 Parameter	2
String Configuration	IO.TAPI.ModemStatus

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

- **"No status!"**: **Modem** Interface was not open yet or was already closed
- **"Modem initialized OK!"**: **Modem** Interface was initialized successfully
- **"Modem error at initialization!"**: Driver could not initialize modem's line. Check Driver's log file for more details
- **"Modem error at dial!"**: Driver could not start or accept a call
- **"Connecting..."**: Driver started a call successfully, and is currently processing that call
- **"Ringing..."**: Indicates that the modem is receiving an external call, but it did not accepted it yet
- **"Connected!"**: Driver connected successfully (completed or accepted an external call)
- **"Disconnecting..."**: Driver is turning the current call off

- **"Disconnected OK!"**: Driver turned the current call off
- **"Error: no dial tone!"**: Driver aborted a call because the available line signal was not detected
- **"Error: busy!"**: Driver aborted a call because the line was busy
- **"Error: no answer!"**: Driver aborted a call because no answer was received from the other modem
- **"Error: unknown!"**: Current call was aborted because of an unknown error

IO.TAPI.PhoneNumber

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

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

Properties

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

IO.TAPI.AcceptIncoming

9 Configure to False if the modem cannot accept external calls (the Driver behaves as a master) and configure to True to enable receiving calls (the Driver behaves as a slave).

IO.TAPI.ModemID

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

NOTE
It is advisable that this property be configured to 0 (zero), indicating that the Driver must use the first available modem.

IO.TAPI.PhoneNumber

A The telephone number used by **Dial** commands. For example, "0w01234566" (the "w" character forces the modem to wait for a call signal).

RAS Interface Configuration

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

I/O Tags

Tags of RAS Interface (N2/B2 = 5)

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

Properties

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

NOTE

The **RAS** Interface uses the **Ethernet** Interface, which for this reason must be also configured.

IO.RAS.ATCommand

A **AT** command to send through a socket to force a RAS device to perform a call using the current RAS channel. Example: "ATDT6265545".

IO.RAS.CommandTimeoutSec

9 Time to wait for a **CONNECT** message in response to an **AT** command, in seconds.

Serial Interface Configuration

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

I/O Tags

Tags of Serial Interface (N2/B2 = 2)

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

Properties

These properties control the configuration of **Serial** Interface.

IO.Serial.Baudrate

9 Specifies a baud rate of the serial port, such as 9600.

IO.Serial.CTSTimeoutMs

9 Time to wait for the **CTS** signal, in milliseconds. After turning the **RTS** signal on, a timer is started to wait for the **CTS** signal. If this timer expires, the Driver aborts sending bytes through the serial port. Available only when the **IO.Serial.RTS** property is configured as **Toggle** and the **IO.Serial.WaitCTS** property is configured to True.

IO.Serial.DataBits

9 Specifies the number of data bits to configure the serial port. Possible values are the following:

- **5**: Five data bits
- **6**: Six data bits
- **7**: Seven data bits
- **8**: Eight data bits

IO.Serial.DelayAfterMs

9 Number of milliseconds to delay after the last byte is sent through the serial port, but before turning the **RTS** signal off. Available only when the **IO.Serial.RTS** property is configured to **Toggle** and the **IO.Serial.WaitCTS** property is configured to False.

IO.Serial.DelayBeforeMs

9 Number of milliseconds to delay after turning the **RTS** signal on, but before data is sent. Available only when the **IO.Serial.RTS** property is configured to **Toggle** and the **IO.Serial.WaitCTS** property is configured to False.

IO.Serial.DTR

A Indicates how a Driver deals with the **DTR** signal:

- **OFF**: **DTR** signal is always turned off
- **ON**: **DTR** signal is always turned on

IO.Serial.InterbyteDelayUs

9 Delay time, in milliseconds (1/1000000 of a second), for each two bytes sent through the **Serial** Interface.

IO.Serial.InterframeDelayMs

9 Delay time, in milliseconds, before sending a packet after the last packet sent or received.

IO.Serial.Parity

A Specifies a parity for the configuration of the serial port. Possible values are the following:

- **E or Even**: Even parity
- **N or None**: No parity
- **O or Odd**: Odd parity
- **M or Mark**: Mark parity
- **S or Space**: Space parity

IO.Serial.Port

9 Number of the local serial port:

- **1**: Uses the COM1 port
- **2**: Uses the COM2 port
- **3**: Uses the COM3 port
- **n**: Uses the COM*n* port

IO.Serial.RTS

A Indicates how a Driver deals with the **RTS** signal:

- **OFF**: **RTS** signal always off
- **ON**: **RTS** signal always on
- **Toggle**: Turns the **RTS** signal on when transmitting data and turns the **RTS** signal off when not transmitting data

IO.Serial.StopBits

9 Specifies the number of stop bits for the configuration of the serial port. Possible values are the following:

- **1**: One stop bit
- **2**: One and a half stop bit
- **3**: Two stop bits

IO.Serial.SupressEcho

9 Use a value different from 0 (zero) to enable suppressing the echo or 0 (zero) to disable it.

IO.Serial.WaitCTS

☑ Configure to True to force a Driver to wait for the **CTS** signal before sending bytes when the **RTS** signal is turned on. Available only when the **IO.Serial.RTS** property is configured to **Toggle**.

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
4.0.19	08/05/2019	M. Ludwig	• Driver ported to Visual Studio 2017 (*Case 27092*).
4.0.17	02/08/2018	M. Ludwig	• Implemented **Int** and **DInt** data types for syntactical parameters (*Case 23837*).
4.0.16	11/28/2017	M. Ludwig	• Implemented rack, slot, and destination TSAP settings for backup addresses 2 and 3 (*Case 23428*).

VERSION	DATE	AUTHOR	COMMENTS
4.0.13	09/11/2017	M. Ludwig	• Fixed a freeze on readings when using simultaneous connections after an unknown time (*Case 23203*). • Implemented a watchdog mechanism to trigger IP address switches (*Case 23270*).
4.0.12	07/06/2017	M. Ludwig	• Fixed a problem when reading **Strings** in **ISOTCP** and **ISOTCP243** protocols with extra connections (*Case 22950*).
4.0.11	06/20/2017	M. Ludwig	• Disabled a grouped reading optimization when configured protocols **ISOTCP** and **ISOTCP243** with optimization for Simultaneous Requests (*Case 22897*).
4.0.10	06/12/2017	C. Mello	• Adjustments to isolate SOE Collecting from Superblock services and callback-oriented simultaneous connections (*Case 22785*).
4.0.9	05/29/2017	F. Englert	• During a reconnection to a backup CPU, now there is an additional check whether the destination TSAP address corresponds to the new IP address. If it is not the expected IP address, this Driver does not send a connection request, and automatically recreates it considering the new IP address and its respective TSAP (*Case 22020*).
4.0.7	05/15/2017	M. Ludwig	• Added TSAP destination settings in hexadecimal format (*Case 22432*).
4.0.4	04/10/2017	M. Ludwig	• Implemented readings and writings of database gateway (*Case 22249*). • Fixed a reading error of single **BOOL** and **BYTE** (*Case 22451*).
4.0.1	03/16/2017	M. Salvador M. Ludwig	• Performance improvements on **ISOTCP** protocol (*Case 22246*).

VERSION	DATE	AUTHOR	COMMENTS
3.1.2	05/12/2015	M. Ludwig	• Fixed a denial on the option to select **ISOTCP243** protocol (*Case 18675*).
3.1.1	09/19/2014	M. Ludwig	• Implemented CPU redundancy (automatic selection of backup CPU, alternative Rack or Slot, with connection to the backup IP address, *Case 15782*). • Implemented configuration of Rack, Slot, and connection type on Driver's properties window (*Case 15911*). • Added Interface-specific Tags for the extra connections option (*Case 17221*).
3.0.1	12/20/2103	M. Salvador M. Ludwig	• Implemented internal Superblocks in extra TCP connections (*Case 14025*). • Driver ported to **IOKit** 2.00 (*Case 14019*).
2.13.1	08/21/2012	M. Ludwig	• Implemented the **PDU REF** field functionality in **ISOTCP** protocol (*Case 13299*).
2.12.1	05/30/2012	C. Mello	• Added support for SOE Collecting of events in DB tables (*Case 12483*).
2.11.1	08/04/2011	M. Ludwig	• Included a consistency according to the **MPI** protocol and code improvements (*Case 12392*). • Added information about supporting PLCs Siemens S7-1200 series (*Case 12292*).
2.10.1	03/25/2011	M. Ludwig	• Implemented the **S7 String** format and a new properties window to configure **Strings** (*Case 12005*).
2.9.1	08/25/2009	M. Ludwig	• Fixed a bug when reading **Counter**-type variables (*Case 10701*). • Implemented advanced configurations for **ISOTCP** and **ISOTCP243** protocols (*Case 10717*).

VERSION	DATE	AUTHOR	COMMENTS
2.8.1	06/19/2009	M. Ludwig	• Fixed a bug in a disconnection addressing multiple slaves in **MPI** protocol (*Case 10595*).
2.7.1	06/03/2009	M. Ludwig	• Implemented the **S5Time** data type (*Case 10413*).
2.6.1	01/07/2009	M. Ludwig	• Fixed a connection failure under **ISOTCP** protocol (*Case 10138*).
2.5.1	11/04/2008	M. Ludwig	• Improvements on properties window layout (*Case 9994*). • Implemented an operation delay in **PPI** protocol (*Case 9968*).

VERSION	DATE	AUTHOR	COMMENTS
2.4.1	04/01/2008	M. Ludwig	• Fixed a problem when addressing analog inputs and outputs combined with the **EnableReadGrouping** property configured to True (*Case 8927*). • Improvements and consistencies to avoid PLC's disconnection problems, as described on *Case 8968* (receiving random values in alarm variables in **ISOTCP** protocol). • Fixed an unhandled exception when receiving NAK characters in **MPI** protocol, which caused a lock on data reception (*Case 8981*). • Improvements on consistency of **MPI** protocol reception (*Case 8981*). • Removed an unnecessary byte in the frame, which caused problems when writing bytes and bits under **ISOTCP** protocol and the S7-400 PLC (*Case 9021*). • Fixed a bug in the automatic reconnection after a physical disconnection in **ISOTCP** protocol (*Case 9030*). • Fixed the implementation of a long ACK frame reception in **PPI** protocol (*Case 9118*). • Implemented a condition of unavailable data in **PPI** protocol. When this condition is met, returns an empty list and OK instead of a failure (*Case 9232*). • Fixed a wrong attribution of Service Access Point in **MPI** protocol, which caused communication failures with Tecnatron adapters (*Case 9238*).

VERSION	DATE	AUTHOR	COMMENTS
2.3.1	09/13/2007	M. Ludwig	• Fixed a reconnection problem with serial adapters when the PLC is turned off (*Case 8069*). • Implemented addressing to multiple slaves in **MPI** protocol (*Case 8625*). • Ethernet port freely configurable (*Case 8683*). • Driver compiled with **IOKitLib** v1.14 to fix reading and writing errors before the first connection (*Case 7614*). • Documentation updated with information about the length of **Strings**, protocols, and compatible devices (*Case 8206*).
2.2.1	03/28/2007	M. Ludwig	• Fixed the lack of creating a blob, which caused errors at run time (*Case 8015*). • Fixed a problem of switching IP addresses at run time (*Case 8026*). • Added support for **Windows CE** (*Case 7504*). • Added support for IBHLink converters (*Case 7994*). • Fixed a writing problem with **Strings** (*Case 7967*).
2.1.1	07/10/2006	M. Ludwig	• Fixed the parsing of DB variables (*Case 7172*).

VERSION	DATE	AUTHOR	COMMENTS
2.0.1	04/13/2006	M. Salvador M. Ludwig	• Fixed a failure in **PPI** protocol Error: Single DLE in data field (*Case 6644*). • Removed address check. Regardless of the data type, any value in *N4* is allowed (*Case 6644*). • Fixed a bug in the configuration interface, where IBHLink converter and **ISOTCP** protocol configurations were mixed. Port 1099 was forced instead of port 102 (*Case 6644*). • Added support for Superblocks and symbolic addressing (*Case 6644*).
1.1.1	11/03/2005	M. Ludwig	• Optimization, standardization, and source code review.
1.0.1	05/01/2005	M. Salvador	• Original version of this Driver.

Headquarters
Rua 24 de Outubro, 353 - 10º andar
90510-002 Porto Alegre
Phone: (+55 51) 3346-4699
Fax: (+55 51) 3222-6226
E-mail: elipse-rs@elipse.com.br

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 CoolVendor

OPC FOUNDATION MEMBER

Microsoft Partner
Gold Independent Software Vendor (ISV)