



Supervision and control XML-based
from Windows to Windows CE

I/O Drivers Reference Guide

Version 11.6 - Ed. oct. 2019

Table Of Contents

| | |
|---|--|
| 1. COMMUNICATION DRIVERS | 1 |
| 1.1. INTRODUCTION | 1 |
| 1.1.1. Introduction to Drivers | 1 |
| 1.1.2. Drivers in Projects | 1 |
| 1.1.3. Linking to memory areas | 3 |
| 1.1.4. Dynamic Addressing | 4 |
| 1.1.5. Task Addressing | 5 |
| 1.1.6. Driver Installation | 6 |
| 1.1.7. Driver Communication Configuration File | 7 |
| 1.1.8. Driver Duplication | 7 |
| 1.1.9. Running Drivers | 8 |
| 1.1.10. Hardware RS232 Errors | 9 |
| 1.2. GENERAL SETTINGS | 10 |
| 1.2.1. General (Drivers) | 10 |
| 1.2.2. Debug | 12 |
| 1.3. STATION SETTINGS | 12 |
| 1.3.1. Stations | 12 |
| 1.3.2. General (Stations) | 13 |
| 1.3.3. Half Duplex | 14 |
| 1.3.4. Serial Port Settings (serial drivers only) | 14 |
| 1.3.5. Queue Size | 15 |
| 1.3.6. Timeouts | 15 |
| 1.3.7. Bridging Service Settings | 16 |
| 1.3.8. TAPI Settings (serial drivers only) | 18 |
| 1.3.9. TCP/IP Settings (ethernet drivers only) | 19 |
| 1.3.10. RAS Settings (ethernet drivers only) | 20 |
| 1.3.11. Special TAPI and RAS configurations | 22 |
| 1.4. TASK SETTINGS | 22 |
| 1.4.1. Tasks | 22 |
| 1.4.2. Static Tasks | 23 |
| 1.4.3. Dynamic Tasks | 25 |
| 1.5. IMPORT DEVICE DATABASE | 28 |
| 1.5.1. Import Device Database | 28 |
| 1.6. API BASIC SCRIPT INTERFACES | 29 |
| 1.6.1. Basic Script Interface Use | 29 |
| 1.6.2. DriverInterface | 31 |
| DriverInterface Events | 31 |
| DriverInterface Functions | 32 |
| DriverInterface Properties | 41 |
| 1.6.3. TaskInterface | 41 |
| TaskInterface Functions | 41 |
| TaskInterface Properties | 48 |
| 1.6.4. StationInterface | 55 |
| StationInterface Functions | 55 |
| StationInterface Properties | 59 |
| 1.6.5. SerialTAPIHalfDuplexBaseStationInterface | 63 |
| Functions | 63 |
| Properties | 66 |
| 1.6.6. SerialStationInterface | 68 |
| SerialStationInterface Functions | 68 |
| SerialStationInterface Properties | 69 |
| 1.6.7. TAPIStationInterface | 76 |
| TAPIStationInterface Functions | 76 |
| TAPIStationInterface Properties | 79 |
| 1.6.8. TCIPStationInterface | 85 |
| TCIPStationInterface Functions | 85 |
| TCIPStationInterface Properties | 87 |
| 1.6.9. RASStationInterface | 93 |
| RASStationInterface Functions | 93 |
| RASStationInterface Properties | 95 |
| 1.7. ERRORS | 102 |
| 1.7.1. Error Descriptions | Errore. Il segnalibro non è definito. |

| | |
|------------------|-----|
| 1.8. ABOUT | 107 |
|------------------|-----|

1. Communication Drivers

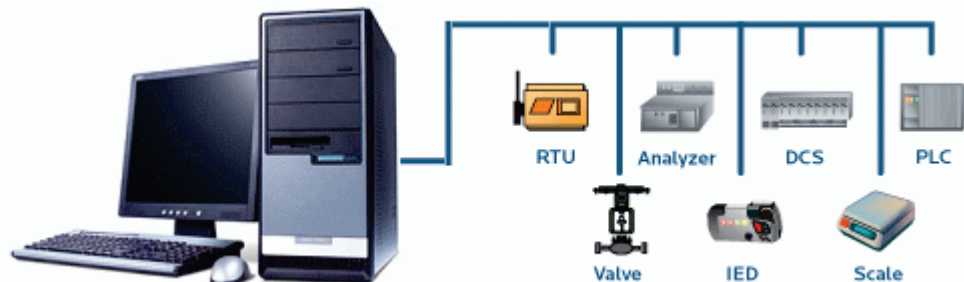
1.1. Introduction

1.1.1. Introduction to Drivers

This section includes a detailed description of the settings to configure in the Supervisor Communication I/O Drivers used in supervision projects. To access the communication drivers settings, select Real Time DB and List Comm.Driver from the project explorer, then right click on the driver name.

The drivers are dynamic libraries (.DLL files) which, based on 'exception-based' logic, transfer the information received from the connected device's memory areas to the Supervisor memory areas and vice-versa, according to predefined settings.

By using the Driver's settings you can specify the associations between the field variables and the Supervisor variables. The system, through the serial port, the fieldbus or the network being used, will read and write the variables 'from' and 'to' the plant, according to the set modalities.



The thread pooling technology, adopted by the Supervisor drivers, is used for exchanging information with the field in the most efficient way, by managing automatically optimized communications according to the effective use of the variables in the project being run.

In fact, only the **effectively-in-use** variables from the system will be exchanged with the field, leaving the driver the job of optimizing and making communications more efficient.

The Supervisor Drivers have been enhanced to make Supervisor communication extremely more powerful and scalable.

Thanks to the common to all driver features, you will be able to get:

1. Optimized and efficient communication
2. Links to PLC addresses that can be managed directly in Tags or indirectly through 'Tasks'
3. Runtime configurability through VBA Script interface
4. Automatic device database importing
5. Bridging function to allow transparent access to external devices via modem (ie. tele-service)
6. TAPI functions to allow automatic calls for remote serial devices via modem
7. RAS functions to allow automatic calls for remote ethernet devices via modem
8. Advanced debugging functions
9. Immediate and direct cable and communication Testing

1.1.2. Drivers in Projects

A supervision project is built with a set of resources and objects and once compiled and processed in Runtime will manage information, logics and user interfaces as required.

Communicating with the field is determined by the usage of Communication Drivers (**as well as the OPC technology**). The Drivers, in forms of dynamic libraries, have the task of reading or writing in the memory areas of the device, connected to the memory areas managed by the project.



For instance, when using PLCs the driver will use the communication protocol provided by the device for exchanging data between the memory areas of PLC and the Supervisor and vice-versa, according to the configurations and data associations set in the driver properties and/or in the variable properties of the Supervisor project.

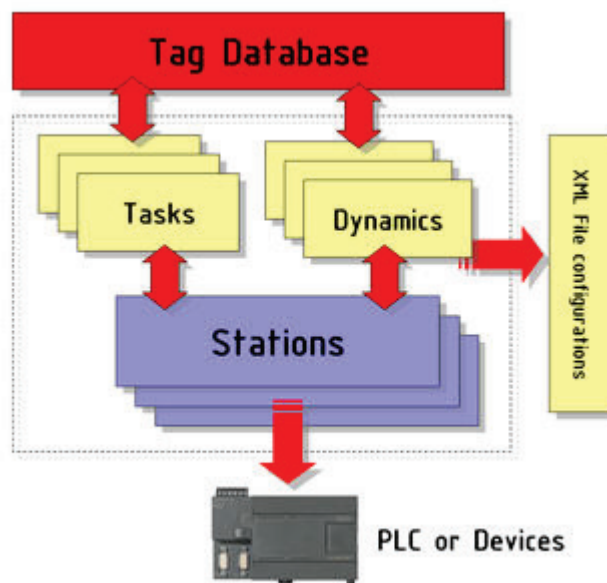
The driver can read or write in the memory areas and the transmission modalities are determined by the devices constructors, therefore it is advised to carefully read the instruction provided with the specific device you wish to connect to.



Carefully read and follow the specifications given by the hardware device constructors for connecting and associating data between the device and the Supervisor.

Independent of the protocol and the hardware constructors, the Supervisor drivers allow the programmer to use the same user interface to configure and setup communications according to the configuration possibilities provided.

The structure of a communication driver is described in the illustrated layout below:



According to the layout above, the driver manages communication protocol at a low level.

1. The driver requires the setting of the main communication parameters by using the **'Station'** concepts (the relative parameters need to be set according to whether you are dealing with serial or network drivers)
2. By means of **"Tasks"** concept, the driver consents 'indirect' association between the device addresses and the variables of the Supervisor project to be setup. The tasks offer the possibility to communicate in data blocks, by setting a variable (or a group of variables) in association to a device's address (or start address). This way the user can configure links to the device's memory areas in indirect mode and therefore independently from the project.
3. By using the **'Dynamic Address'** concept, the driver consents a direct association of the memory address in the project's Tag properties. In this way the variable points directly to the devices address, leaving the driver the job of dynamically creating communication tasks which will always be optimized.
4. The driver always works with the Supervisor project's **"Real time Database"**. The variables are therefore associated directly (Tags property) or indirectly (Tasks). In any case communication is optimized according to the 'Variables in use' concept.
5. The driver's configuration is saved in the appropriate **XML files** in the project's 'Resources' folder. The files are based on XML meta-language, as with all the project, for maximum transparency. The driver's files are:
 <driver_name>.drvsettings = files containing the driver's general settings
 <driver_name>.dynsettings = files generated upon runtime start with the characteristics of the calculated dynamic tasks.

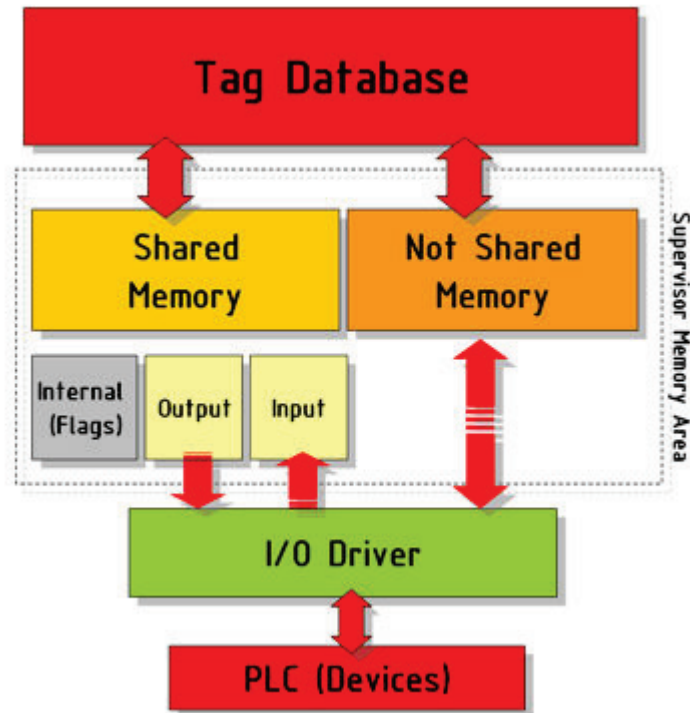


The Supervisor Drivers are libraries which can also be used independently of the Supervisor projects, that is, they can also be inserted in other programming environments which are compatible with the ActiveX technology.

1.1.3. Linking to memory areas

The Supervisor Real Time DB tags are dynamically mapped in the PC's memory, independently from the communication techniques used. The mapping in the Supervisor Tags' memory areas is setup in the 'general properties' of each single tag.

The variables can be mapped in two different data areas, the data area defined as '**Not Shared Memory**' (proposed for default) and the data area defined as '**Shared Memory**'.



'Not Shared' Supervisor Area

The 'Not Shared' data area is the area proposed as default when creating Tags in the Supervisor project. When the 'Not Shared' areas are used, the Supervisor will decide how and where its Tags will be allocated in memory.

In this way the user does not need to worry about allocating the tags and where the Supervisor will allocate them at runtime. The user should only specify addresses for links to devices, if any.

Usually this selection is preferably used to avoid allocating incorrect data internal the Supervisor. The Supervisor automatically manages its information internally and only communicates with the driver when the variable is connected to the device.

'Shared' Supervisor Area

The 'Shared' data area, when selected, permits you to specify data allocation inside the memory areas of the Supervisor, independently of the communication driver. **When selecting this area the programmer is required to assign the area and absolute address of the Tag in the Supervisor's memory.**

Therefore the programmer must be very careful in assigning the correct address to avoid superimposing other variables in memory by mistake.

The Shared area requires the selection of the **Area Type** by choosing one of the following:

1. **Input Area**
2. **Output Area**
3. **Flag Area (Internal)**

Shared Area variables can be exchanged with the field using Input, Output or Input/Output independently from the type of area selected.

Link to device

A task defines the link between a group of Supervisor's tags and the corresponding memory area of the device, also specifying the methods to access and manage data.

The Supervisor offers you the possibility to set the driver's communication using two different task concepts: the 'static' task (usually defined simply as 'task') and the 'dynamic' task.

Dynamic Tasks: these Task are automatically created by the driver at project startup, based on the links to device's addresses set in the 'Dynamic Address' properties of each single tag. The driver will manage tags grouping and performances optimization.

Static tasks: these Task are defined by the designer who has the responsibility to define grouping and communication parameters.

1.1.4. Dynamic Addressing

The Supervisor permits the device's address to be specified directly in the Tag's general properties, in the project's Real Time Database resource. When using this technique the driver generates communication tasks in dynamic mode, according to the optimization and grouping concepts of the predefined data. At project startup the driver will generate a number of tasks by grouping data blocks and will activate communication only when the variables are used in the project.

To assign the address in dynamic mode, you need to:

1. Select "Dynamic" from the tag's general properties window
2. Select the Comm. Drivers tab from the Tag Browser window
3. Double click on the desired driver in the list. The 'Task property' window for assigning the address will appear
4. Specify the address in the 'Device address' field
5. Specify the data management (read, write or both) in 'Type' field

The tag's dynamic address can also be specified by directly typing it in the 'Dynamic' property edit box. The address syntax is:

[DRV]<Name of Driver>.Sta=< Name of station>|Addr=< Address of device>



Please refer to the specific addressing documentation of each Driver in order to use the appropriate syntax.

Dynamic Task Concepts

As mentioned above, the Supervisor will dynamically create a number of tasks to manage the communication at the project startup.

The "**Minimum Threshold**" parameter, in driver's general properties, is a basic parameter used to determine the automatic task generation. This parameter sets the minimum value as the fragmentation threshold in generating a number of tasks.

For instance:

VAR00001 is linked to the device's word 0 address
VAR00002 is linked to the device's word 3 address
VAR00003 is linked to the device's word 12 address
VAR00004 is linked to the device's word 18 address

The 'Minimum Threshold' parameter is set to 5 by default. In this example when the project starts, the driver dynamically creates the '.dynsettings' XML file in the project's 'Resources' folder, grouping the tags in 2 dynamic tasks.

In the first task the words from 0 to 3 are read, in the second task (needed because the address of the next word to be read exceeds 5 bytes distance as indicated by the 'Minimum Threshold' parameter) the words from 12 to 18 are read.

If, however, we set the 'Minimum Threshold' parameter at 20, the driver will dynamically create one task only by reading from the word 0 to the word 18 at project startup.



The number of tasks generated automatically depends on the value set in the 'Minimum Threshold' and 'Aggregation limit' parameters.

The tags associated to the driver in 'dynamic' mode will be automatically managed by default in read-write mode by the driver. The driver will decide whether to manage only in write or in read-write mode according to the possibilities offered in the associated device's area. Moreover this setting can be

changed in the 'Task property' window or through the '.dynsettings' XML file generated automatically at the start of the project run. The data in the XML file regarding run type is as follows:

<TypeName> where it is possible to change the default value 1 with the following values:
0 = Input (read only)
1 = Input/Output (read/write)
2 = Output on exception (write only if value has changed)
3 = Output continuous (write anyway)

1.1.5. Task Addressing

The Supervisor permits the association between the device's addresses and the project's Real Time Database tags' addresses in an indirect mode using 'static' tasks. By using this technique, the driver will require the association between a set of project tags and the correspondent device's memory areas to be specified in the configuration properties of each single Task. In addition to this the type of communication also needs to be set (read, write, or both).

The list of generated tasks will be saved in the "drvsettings" XML file and therefore will remain there independently from the project.

To create and configure a task, you need to:

1. Select the 'Tasks' card from the driver settings
2. Insert a new Task using the 'Add' button
3. Configure the properties of the task according to the data exchange requirements
4. Confirm with OK and proceed with the next task if needed



Important Note: to set a list of tags in the task which should be consecutive starting from the device address, type in the 'Variables' field the names of the tags, separated by the ';' character. The ellipse button helps in selecting the tags from the Real Time database tags' list.

A task created with this procedure can include one or more tags, all of the same data type (i.e all word or all float) and consecutive in device memory area. The address to specify for the task is the starting address, corresponding to the first tag in the list.

Only a few drivers allow task creations which include different types of variables (i.e. Byte, Word, etc). For further information, please consult each driver's manual documentation.

The following Task execution principles are to be kept in mind:

| | |
|---------------------|---|
| Input | This kind of task reads data from the connected device and writes it in the Supervisor tags. They can also be executed on event. In this case the task will be executed only when the associated tag is set different from 0. When the Input task is not executed on event, it will be executed with the polling technology together with the other input tasks. |
| Output | This kind of task writes the Supervisor Tag data values in the connected device. The output tasks are executed by the Supervisor with the 'Event-driven' technology, which means only when there is a data variation in the Supervisor which needs to be notified to the PLC or to the connected device. The driver also permits continuous data writing. |
| Input/Output | The input/output tasks are managed in 'polling' mode to keep data read from the connected device updated, whereas the writing of data is performed on event only when there is a data change in the Supervisor, by rewriting the changed data to the connected device. The input/output tasks, however, always read data from the device first, then write data when necessary. |
| COM (OLE2) | The Supervisor can also manage tasks which have not been configured directly in the driver, but performed at runtime by the VBA Basic Script. In this case the tasks are executed by the driver according to script code written by the user, by executing the read or write tasks in synchronous or asynchronous mode. |

Even though the programmer must be very careful when using Tasks, he/she will however have the benefit of being able to decide how data is to be exchanged and not the Supervisor.

The communication tasks are executed by tag name only and not by absolute address.



Remember that the tasks also allow the COM (*Component Object Model*) driver interface to be used for handling or performing any communication task using **VBA scripts**.

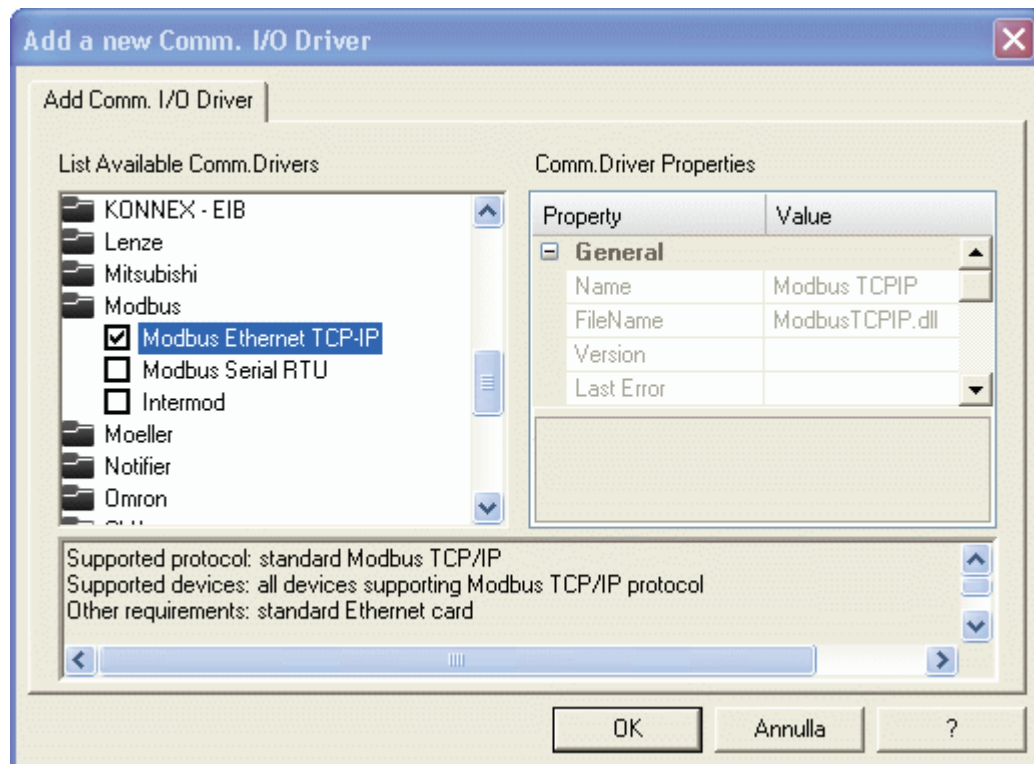
1.1.6. Driver Installation

The Supervisor installation provides the automatic installation of the available drivers library together with the platform.

The Supervisor Drivers, being built as simple .DLL files, can be easily enlarged or updated independently from the development platform. To update or install a new driver, just copy the relevant .DLL file into the Drivers folder, which is found inside the Supervisor installation folder (I.e. C:\Program Files\Progea\Movicon11.3\Drivers).

Project designers may choose which communication driver to insert and configure in their various applications, simply selecting it from the list provided.

The insertion and setting up of a communication driver is done in Supervisor programming mode, through the 'Comm. Drivers List' resource from the 'Real Time DB' group in the 'Project Explorer' window. When activating the 'Add new Comm. Driver' command a dialog window, containing a list of the drivers available, will display.



The bottom window shows important data linked to the selected communication driver:

- Brief protocol description
- List of supported devices
- Cards or libraries needed
- Any driver restrictions: areas not supported, connections with more than one PLC not possible, etc.

Once the driver has been inserted, it can be configured through the Supervisor 'Properties Window'. More than one communication driver can be inserted into a single project, as long as they comply with the options set on the hardware key.

A new item called the ""Renaming Manager" has been added the Communication Drivers list of features which allows you to know whether the driver supports the renaming management. Those drivers that support this management, option set to 'true', consent variables to be displayed with names modified within their configuration window; and to support apply new name command.

1.1.7. Driver Communication Configuration File

When a new communication driver is added to the project, it can be setup to load any existing custom ".drvsettings" configuration file. In this way the driver will be able to obtain the settings initially set by the programmer if different from those for default.

While the Visu+ is being installed all or some of the subfolders indicated in the Visu+ installation folder will be created and within which there should be driver "drvsettings" configuration files:

```
[Install Folder]\drvsettings\WinCE\  
[Install Folder]\drvsettings\Win32\  
[Install Folder]\drvsettings\ClientCE\  
[Install Folder]\drvsettings\ClientWin32\  
[Install Folder]\drvsettings\ClientJ2SE\  
[Install Folder]\drvsettings\ClientJ2ME\
```

As you can see there are six subfolders, one for each platform that can be selected in the project. In this way different configuration files can be defined for the same driver according to the platform type selected. If any of the folders listed above do not exist, they can be added manually.

When a communication driver is added to the project, Visu+ will search for an existing file with the same driver name and ".drvsettings" extension (i.e. "ModbusTCP/IP.drvsettings") within the folder corresponding to the platform selected for the project. If and when the file is found it will be copied to the project resource folder (the file's date and time will be set with the current date and time). If the file is not found the driver settings window will open using the default settings.

For instance, if the project has been set with the "Windows CE" platform, and the "Modbus TCP/IP" driver is added, Visu+ will check whether a file named "ModbusTCP/IP.drvsettings" exists in the "[Install Folder]\drvsettings\WinCE\" folder, and if and when found it will be copied to the project resource folder.

If more than one platform has been enabled in the project, the order of priority with which the ".drvsettings" files will be selected for copying is: WinCE, Win32, ClientCE, ClientWin32, ClientJ2SE, ClientJ2ME.

In cases when the ".drvsettings" file already exists in the resource folder when inserting a driver, this file will be left unchanged and therefore not overwritten.

The configuration files created while installing Visu+ may be modified as required.

1.1.8. Driver Duplication

The Supervisor provides the possibility to install two or more drivers in the system which can be selected from the list of those available as seen in the previous paragraphs. However, if you wish to install two or more identical drivers (for PLCs or devices of the same type), you will need to follow the simple indications as described below.

For instance, you may need to install two or more drivers of the same type to communicate in the same system environment at the same time but it is not enough to simply set two different Stations with the same protocol for the device of the same type.

Example: you want the PC, on which the Supervisor is installed, to communicate with two identical devices on two distinct PC communication channels. To duplicate the Communication Driver you need to:

1. Duplicate the driver's DLL file which you can find in the "Drivers" folder in the Supervisor installation folder. The copied file must have a name different from the original one, for instance the same name plus an increasing index.
2. Create in the "Drivers.xml" file a new block indicating the new .DLL file, just by duplicating the block of the original file and giving it a new name and description, so that they appear as two identical drivers but with different names and descriptions on the Supervisor Driver list.



When a driver is duplicated it will result and have all the effects of a new driver, therefore you will need to provide an additional driver in the license options. When using two drivers, even though of the same type because duplicated, the license must be enabled for two communication drivers.

A practical example:

You wish to manage a project where two "Modbus Serial" drivers must be used. The following steps must be carried out:

Dll Duplication

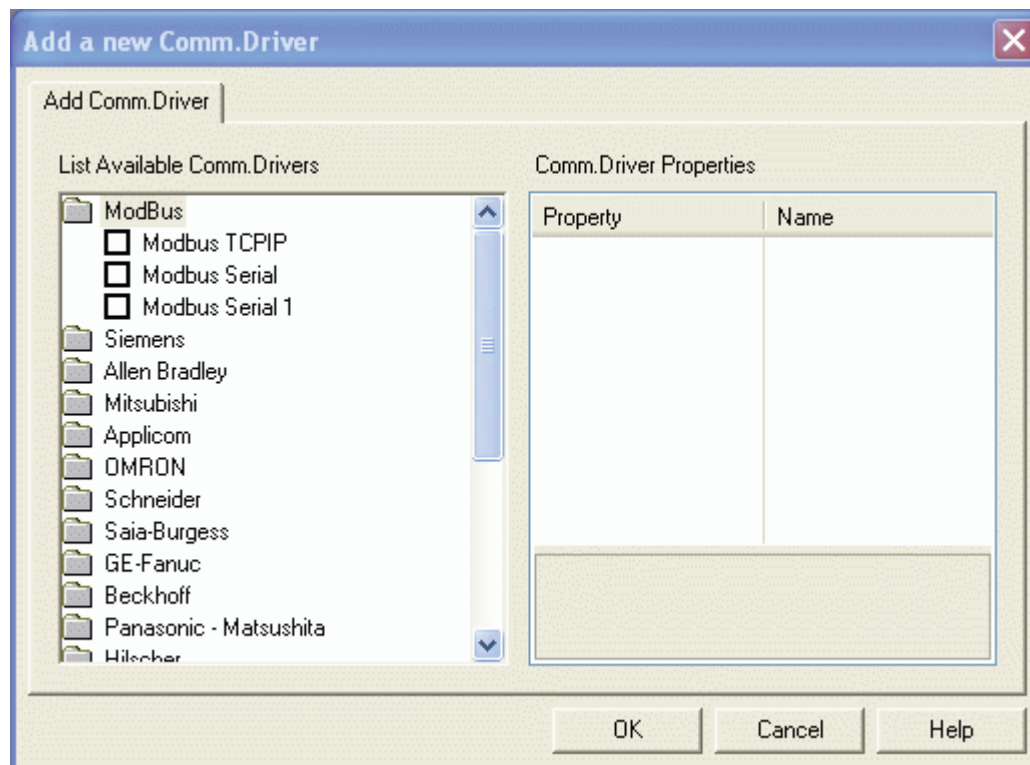
Make a copy of the original **"Modbus.dll"** in the Supervisor Installation "Drivers" sub-folder and call it i.e. **"Modbus1.dll"**.

Drivers.xml file Modification

Open the "Drivers.xml" file, located in the Supervisor installation "Drivers" sub-folder, with a text editor. Add the new driver in the <DriverFactory Factory="ModBus"> tag, by specifying a description as pleased and the name of the new dll. For example: <Driver Name="Modbus Serial 1">ModBus1.dll</Driver>. Modified file should result as:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DriverList>
  <DriverFactory Factory="ModBus">
    <Driver Name="Modbus TCPIP">ModBusTCPIP.dll</Driver>
    <Driver Name="Modbus Serial">ModBus.dll</Driver>
    <Driver Name="Modbus Serial 1">ModBus1.dll</Driver>
  </DriverFactory>
  ...
  ...
```

In this way, when you open the window with the list of available Drivers, you will also find the one you have just duplicated.



1.1.9. Running Drivers

The installed communication driver or drivers will be run at the application project's startup and will remain active for the whole duration of the project's processing, according to the execution modes chosen or the VBA scripts logic using the driver COM (Component Object Model) interface. At each activation of the communication with the field, the system will record a communication status notification message in the Historical Log.



The presentation of a green coloured led on the Status Bar on the Supervision window's bottom border (if displayed) means that the installed driver is

communicating correctly with the field. The red coloured led indicates that there is a communication error.

Any communication problems (cable, connections, settings, etc.) will generate communication errors that will be alerted by the driver in the Status Bar and recorded in the Historical Log.



Note: the drivers are independent of the project, and their configurations are saved in separate appropriated files, which are identified by the ".drvsettings" and "dynsettings" extensions. This philosophy guarantees that the project is kept intact when changing PLCs or communication devices.

The driver can be subordinated to the conditions established by the programmer during runtime execution.

1.1.10. Hardware RS232 Errors

The Supervisor Communication Drivers auto-diagnosis emits codes of hardware communication errors, according to the indications supplied by the recording status of the UART chip of the serial port installed on the PC.

We recommend to use serials ports with UART 16550A chips which use FIFO16 byte data management. The type of serials installed on the PC is easily detected by running the Microsoft Diagnostics MDS.EXE file.

The communication hardware errors are generally due to the following possible causes:

- Serial line disturbances
- Potential difference between the device masses.
- Serial cards inadequate for the performances required
- Defective or inadequate communication cables
- Baud rate too high for the hardware being used
- Communication device breakdown

The hardware error messages supplied by the driver relate to the codes emitted on the error register of the serial UART chip.

When a generic hardware error occurs, the Supervisor reports a number, translated in binary, you can use to identify the error or errors by confronting each single bit whose meanings are described in the table below:

| VALUE | CODE | MEANING |
|-------|-----------|--|
| 1 | RX OVER | The serial has received more characters than the buffer capacity can hold |
| 2 | OVERRUN | The serial has received a character before the previous one was processed by the system |
| 4 | RX PARITY | Error in parity, inconsistency between the parity received and the one set. |
| 8 | FRAME | Data frame error. The data received does not respect the set characters (length, stop bit, etc.) |
| 16 | BREAK | Break Status requested by participant |



All the other communication errors depend on the specified protocol, therefore you need to refer to the indications of that specific driver. The driver error messages are displayed on the status bar and can be viewed in the Historical Log.

1.2. General Settings

1.2.1. General (Drivers)

Some of the properties common to all the communication drivers can be configured on this setting card.

It is not generally necessary to change any of the default settings.

Wait Time

The pause time, expressed in milliseconds, between the execution of two tasks (data blocks) of successive communication. It may be necessary to change the default value for devices which need a wait time between one interrogation and the next (i.e. devices with poor performances). It can be useful to modify the default value also when it is necessary to lighten the workload of the CPU.

Time-out

The time-out time for executing synchronized tasks. The value is expressed in milliseconds.

Minimum Threshold

This parameter determines the minimum threshold for the fragmenting of data blocks exchanged with the device. The Supervisor automatically calculates (at project startup) the size and quantity of dynamic tasks to be created by the driver for dynamic variables (tags with dynamic addresses) communication.

The Supervisor, in fact, tries to optimize the communication grouping the highest number of data in the same task. When the data is linked to addresses distanced between each other, this value determines the distance in bytes which allows the Supervisor to decide whether to create a new task for the next data block.

Example:

VAR00001 is linked to the device's word 0 address
VAR00002 is linked to the device's word 3 address
VAR00003 is linked to the device's word 12 address
VAR00004 is linked to the device's word 17 address

The driver's 'Minimum Threshold' parameter is set to 5 as default. In this example the driver will dynamically create the 'dynsettings' XML file in the project's 'Resources' folder, and the driver will generate 2 dynamic tasks.

In the first task the words from 0 to 3 will be read, in the second task (which is necessary because the next word to be read is to an address which exceeds 5 bytes as indicated by the Minimum Threshold' parameter) the words from 12 to 17 are read.

If, however, we set the 'Minimum Threshold' parameter at 20, the driver will dynamically create one task only by reading from the word 0 to the word 18 upon project startup.



The number of tasks automatically generated depends on the value of 'Minimum Threshold' and 'Aggregation Limit' parameters.

Aggregation Limit

This parameter allows you to specify the maximum number of bytes to be aggregated for each dynamic task. By leaving this parameter to zero the driver will use the maximum value set in the selected protocol as a maximum limit. Changing this parameter value may be necessary when using devices which have a lower maximum byte number limit compared to the protocol being used for exchanging tasks.

Synch.Startup

This option determines the synchronization between the logic and driver communication at the project's startup.

When this option is set to 'True', the Supervisor will wait until the static input tasks have been completely executed before processing the project's logic and scripts.

Though this option will cause the project to take longer in starting up, the logic will run with 'updated' input values ensured.

VBA Interface

Indicates whether the driver supports the COM (Component Object Model) interface. The COM interface, also defined as OLE2, grants the use of VBA script logic for handling the driver, according to the methods, the properties and the events described in the appropriated documents.

Polling Time

This parameter, expressed in milliseconds, represents the minimum polling time for executing tasks for updating data when variables are in use.

This value will be obtained by all the Dynamic tasks with the same Polling Time (Not all drivers allow you to specify a Polling Times for each Dynamic task). As regards to Static tasks, this value is inserted as the default value in the task's property ("Polling Time") when created. This value can then be changed through the Static Task's property afterwards.

Setting this value to 0, meaning that the data is updated with the highest velocity possible.
A higher value can be set, for example, when the data does not require rapid updating times.

Unused Polling Time

This parameter, expressed in milliseconds, allows data updating to be forced even **when the tags are not in use, establishing, however, a polling time.**

This value is obtained by all the Dynamic tasks, which all have the same Unused Polling Time. However, this value is inserted in Static Task properties ("Unused Polling Time") as the default value when created. This value can then be changed through the Static Task's property afterwards.



When this parameter is set to 0, the tasks will not be executed when the tags are not in use.

Error Polling Time

This parameter, expressed in milliseconds, determines the polling time of a station (device) when it is in error; i.e. when there is a station communication error, its tasks are no longer performed with this number of milliseconds. Please note that, in cases of errors, all the station's tasks are suspended (no more will be performed) except the one causing the error.

Protocol Priority

This box is used for setting the communication thread priority, which is the priority given to the driver's execution in respect to other Supervisor processes.

The possible values, starting from low to high priority, are:

- **Normal**
- **High**
- **Very High**
- **Real Time**



Take extreme care not to exceed the CPU workload (CPU at 100%), when changing the default value (Normal).

It may be useful to try and increase the driver's process priority due to elevated communication loads. In this case it is advised to set the Wait Time parameter at a value other than 0, to avoid too many system resources being used.

Suspend tasks in case of error

This property will activate only for those drivers that allow symbolic Task addressing, "Tag Name" and will remain deactivated with 'True' value, for those drivers that allow only numeric addressing, "Address Type".

When setting this property to "True", when a communication error is verified all the tasks will be disabled except the job generating the verified error. This behaviour may result inadequate in cases of drivers that, such as Beckhoff TwinCAT or Allen-Bradley Ethernet/IP, consent there use of symbolic addressing (device variable names). In this case this option can be set to 'False', allowing all tasks to be kept active even when errors occur. The following will happen when setting "Suspend tasks in case of error" to "False":

The driver will carry on with the next interrogations even in cases where interrogations are unsuccessful

Unsuccessful interrogations will be further attempted after the time indicated in the "Error polling Time" parameter has expired

Any communication errors will however reflect on the communication bit status (_SysVar_:CommDriverStatus)

Management of "In Use" state for structures

This parameter, if set to True, consents all members of a Structure variable in use even though only part of it is effectively in use in the project. This will ensure that all of the structure variable members are exchanged by the driver even when only one of them is in use in the project. When setting this property to "False" the "In Use" state will be managed for each structure member singularly, therefore only those members effectively in use in the project will be exchanged by the driver.



CAUTION! Setting this property to True might effect communication performances especially project has loads of structure variables containing a considerable number of members.

Direct Output for Input/Output

If the property is set to 'False' (Default value), when a Supervisor's Variable's value, associated to a "Input/Output" task, is modified, the linked task is performed first in Input mode and then in Output mode only if the device tag has not changed since it was last read. This means that if the variable

value has changed both in the Device and in the Supervisor, the driver will give priority to Device tag value. This behaviour can be modified by setting the "Direct Output for Input/Output" property to 'True', so that the changed Supervisor value gets written directly to the Device Tag. This behaviour is applied to all Tasks of "Input/Output" types.

1.2.2. Debug

Some of the properties concerning the Debug, common to all the communication drivers, can be configured on this setting card.

It is not usually necessary to change any of the default settings.

Debug Window

Default value = False. When selecting 'true', the driver shows all the diagnostic and debug messages generated by the driver in the appropriated Debug window (they can also be viewed in the Supervisor workspace Output bar).



The debug window activation may slow down the task activation and, in general, the communication speed. We suggest you use this option only when strictly needed, such as in the debug phase.

Max Entries

Default value = 10000. The value, relating to the maximum number of diagnostic message strings displayed in the debug window before being recycled, is set in this box. When the default value is left, the window will keep the last 10,000 message strings displayed.

Enable Log to file

Default value = True. If "True", the driver will record on file all the Debug's diagnostic messages generated by the driver.

The amount of recorded data is defined in Max Entries setting, while the file name and path are set in the Log FileName property.

Log FileName

If no file name is specified here, the driver's diagnostic Log file name will be 'System.log' and it will be located in ProjectFolder\LOGS folder. To locate it elsewhere, please specify in this box the file name and path for the driver's diagnostic Log.

The generated file will be a standard text file.

1.3. Station Settings

1.3.1. Stations

In this page you need to insert and configure the '**communication stations**'. Each driver should have at least one communication station inserted and configured.



Please remember that it is possible to set up the driver communication both by using 'dynamic tasks' or 'Tasks' concepts (described in the appropriate chapters). The dynamic tasks are created automatically by the driver upon project startup, using the links to the device's addresses set in the 'Dynamic Address' properties of each single tag.

The communication stations allow you to set, like the driver, how communication must be managed, where each station represents a communication channel towards the configured device.

Add

The "Add" button allows you to insert a new 'station' for the communication driver. When inserting a new station a window will automatically display for setting the parameters of the communication required.

Once the station or stations have been inserted you can change or remove them by using the buttons indicated below.

Edit

The 'Edit' button allows you to modify the parameters of a previously inserted station. Therefore you first need to select the station you wish to modify then use the 'Edit' button or double click on the station name in the list.

Remove

The "Remove" button allows you to delete a previously inserted station. Therefore you need to first select the station you wish to modify then use the 'Remove' button.

Test Cable/Comm.

This button allows you to run a communication test with the device. Thanks to this very handy functionality the driver will attempt communication to verify whether the cables have been connected correctly and the main communication parameters have been properly set.



The communication test is invoked by reading specific data according to the test criteria of each single protocol.

Therefore it might be necessary to refer to the specifications indicated by each of the drivers to see what kind of test is carried out.

For instance, with generic protocols such as Modbus, the test invokes the reading of a specific Function Code (FC2) which may not necessarily have been implemented on the connected device.

Please be reminded again that in each case the test only verifies whether the cable and general parameter configurations have been done correctly, while the correct association between data in the Supervisor and the device is demanded to the project designer whether data in the Supervisor and the device have been associated correctly.

1.3.2. General (Stations)

This settings card is used for defining the settings for the selected station in the 'General' properties group .

Station Name

Name which identifies the station corresponding to the device with which it is to communicate. The station name is the one internal the driver.

When more than one station is to be set, each one must have its own name.

Error Threshold

When there are any communication errors, this parameter sets the number of errors to be reached in order for the Driver to effectively give communication error notification. The internal counter will not alert any occurrences of communication break-down without re-attempting to retrieve communication beforehand. When the number of attempts have been reached the driver will give out an error warning.

State/Command Variable (only managed for drivers using driver base library build 250 and later)

By assigning the name of a numeric variable of the Supervisor (Byte type recommended) to this property, it is possible to check the communication state with the selected station, to enable/disable the communication with this station, to start/stop TAPI connections (serial drivers only) or RAS connections (Ethernet drivers only), and to switch between TCP/IP servers (Ethernet drivers only).

See the following table for the meaning of the variable bits. Please note that some bits can only be used to check the state of the station, while other bits can only be used to set the state of the station (commands).

- The variable's bit 1 can be used to check and modify the station's Active/Inactive state
- Bits 4, 5, 6 can be used to manage a TAPI connection (serial drivers) or a RAS connection (Ethernet drivers)

A "State/Command Variable" of type bit can be used if the only information needed is the communication state (bit 0).

| | |
|----------------------------------|---|
| Bit 0 (State) | Communication State: 0 = OK 1 = Error |
| Bit 1 (State/Command) | Station State: 0 = Active/Enable 1 = Inactive/Disable |

| | |
|----------------------------|---|
| Bit 2 (Command) | Switch the active TCP Server |
| Bit 3 (State) | Active TCP server: 0 = Default Server 1 = Backup Server |
| Bit 4 (State) | Modem connection state: 0 = disconnected 1 = connected |
| Bit 5 (Command) | Open modem connection |
| Bit 6 (Command) | Close modem connection |
| Bit 7 (Unused) | Always 0 |

Keep Opened

This property is present in serial drivers only. It allows you to establish whether the driver must keep the communication port open (and therefore always busy) or not. When the value is set to True, the driver is loaded at the start up of a project run and always keeps the associated communication port open (busy).

When setting the value to False, the driver closes the communication serial port after every 'Input' or 'Output' operation has been done, thus leaving the port free.

1.3.3. Half Duplex

In this group of settings you need to set the configurations inherent to the switching of direction of communication in case of a half duplex serial line.



These settings are normally reserved for the expert user and therefore we recommend that you leave the default settings.

Signal

Selects the signal of the RS232 used to control the direction of transmission. Possible values: **None** (default value), **RTS** e **DTR**.

Delay (ms.) Before First Char

Defines the delay time after the set to ON control signal before giving transmission starts. Default value = 0.

Delay (ms.) After Last Char

Defines the expected time after the transmission of the last character before putting OFF to the control signal. Default value = 0.

The time is expressed in milliseconds.

1.3.4. Serial Port Settings (serial drivers only)

In this group of settings you need to set the configurations inherent in the Serial Port Settings properties group for the selected station.

The serial is managed by the modem's driver when the station's TAPI settings are used for managing communications via modem. The settings with which the serial is opened are therefore those defined in the modem's driver's advanced properties and may be different to those set for the Movicon driver station ("Serial Port Settings").

Port

Default value = 1. Sets the number of the serial port to be used for communicating. I.e. value = 1 for COM1



Note: you need to make sure that no conflicts occur in Windows when using the ports. For instance, when installing the Com4 port, you need to make sure that the assigned address and the IRQ are compatible with the PC's configuration. In order to do this we advise you use addressable serial cards.

Baud rate

Default value = 9600. Sets the velocity of the serial communication (Baud Rate). The value of the communication's velocity must correspond to that of the device to be communicated with.

Byte Size

Default value = 8. Sets the amount of bytes required by the communication protocol in use.

Parity

Default value = Even. Sets the parity type required by the communication protocol in use.

Stop Bits

Default value = 1 stop bit. Sets the number of Stop Bits required by the communication protocol in use.

Flow Control

Default value = None. Sets the data Flow Control type for the type of communication in use. This property permits the flow of communication data from the connected device's serial port to be adapted to low level protocol requirements. The default value, 'None', means no flow control, nevertheless it might be necessary to select a flow control type (i.e. when signalling errors with code '1').

The options are:

None: No Flow Control. Control not required by protocol.

Hardware: Flow Control is managed by electric serial line electric signals (i.e. RTS, CTS, ecc.).

Xon/Xoff: Data Flow Control is Xon/Xoff type.

NONE (Signal disabled): Flow Control set to NONE and serial is set for disabling the DTR and RTS signal management.

RTS Toggle: Sets the serial for managing RTS signal control in Toggle mode, meaning the serial keeps signal high when there are characters to be sent.

1.3.5. Queue Size

In this group of settings you need to set the configurations inherent to the buffer size of the selected station's serial port.



These settings are normally reserved for the expert user and therefore we recommend that you leave the default settings.

Rx Queue

Sets the quantity of data bytes to be managed by the serial port to buffer values being received. The default value is used if not specified differently.

Expert users can change this value to adapt to the needs of the system being used.

Tx Queue

Sets the quantity of data bytes to be managed by the serial port to buffer values being transferred. The default value is used if not specified differently.

Expert users can change this value to adapt to the needs of the system being used.

1.3.6. Timeouts

In this group of settings you need to set the selected station with the configurations inherent in serial communication time-outs.



These settings are generally reserved for expert users and therefore it is advised to keep the default settings.

Rx Timeout

Default value = 5000. Sets the wait time interval value, in milliseconds. When exceeded, the driver notifies time-out of communication reception.
The time-out refers to data being received.

Tx Timeout

Default value = 5000. Sets the wait time interval value, in milliseconds. When exceeded, the driver notifies time-out of communication transmission.
The time-out refers to data being transmitted.

The following parameters apply to serial driver only:

CD Timeout

Currently not in use. Reserved for future use.

CTS Timeout

Default value = 5000. Sets the wait time interval value, in milliseconds. When exceeded, the driver will notify communication time-out for the CTS serial parameter.
Sets the time within which each individual write operation must be completed, at low level (Window API) in the serial port.

DSR Timeout

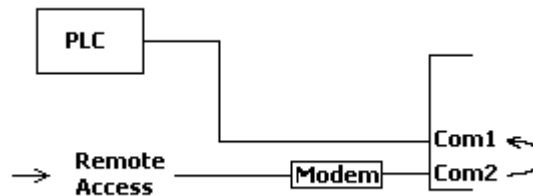
Default value = 5000. Sets time-out value in milliseconds which will also be notified to the DSR serial parameter by the driver. This property sets the maximum time-out between the reception of one character and the next, which will also be the maximum time within which each single read operation must be completed at low level (Windows API) in the serial port.

1.3.7. Bridging Service Settings

In this group of settings you will need to set configurations inherent to the Bridging function of the selected station.

The Bridging functions let the user opt to use the Supervisor as a 'bridge' for tele-services rendering it transparent for any remote communication through modem ports from PC to PLC, meaning that the PLC can be accessed directly from a remote PC by using the driver communication port.

The illustration below demonstrates this type of connection:



The driver will interrupt its communication during access in 'external bridging'.

Enable

Enables the Bridging Function. When set to True, the driver attends to the specified serial port: when a remote call is made (i.e. by tele-service), the driver disconnects the Supervisor from the PLC and connects the modem port with the port connected to the PLC, in transparent mode, until disconnected by command automatically restoring the Supervisor's communication.

The service is disabled by default.

Port

Here you need to specify the number of the COM serial port the modem is connected to and which is to be used for the Bridging service.



Note: you need to make sure that no conflicts occur in Windows when using ports. For instance, when installing the Com4 port, you need to make sure that the assigned address and the IRQ are compatible with the PC's configuration. In order to do this we advise you use addressable serial cards.

Baudrate

Sets the velocity of the serial communication (Baud Rate). The value of the communication's velocity must correspond to that of the device to communicate with.

Byte Size

Sets the amount of bytes required by the communication protocol in use.

Parity

Sets the parity type required by the communication protocol in use.

Stop Bits

Sets the number of Stop Bits required by the communication protocol in use.

Flow Control

Sets the data Flow Control type for the type of communication in use. This property permits the flow of communication data from the connected device's serial port to be adapted to low level protocol requirements. The default value, 'None', means no flow control, nevertheless it might be necessary to select a flow control type (i.e. when signalling errors with code '1'). The options are:

None: No Flow Control. Protocol does not require this control

Hardware: The Flow Control is managed by the serial line's electrical signals (eg. RTS, CTS, etc.)

Xon/Xoff: The data Flow Control is Xon/Xoff

Rx Queue

Sets the quantity of data bytes to be managed by the serial port to buffer values being received. The default value will be used if not specified differently.

Expert users can change this value to adapt it to the needs of the system being used.

Tx Queue

Sets the quantity of data bytes to be managed by the serial port to buffer values being transferred. The default value will be used if not specified differently.

Expert users can change this value to adapt it to the needs of the system being used.

CD Timeout

Currently not in use. Will be implemented in the near future.

CTS Timeout

Sets the time-out value in milliseconds which will also be notified to the CTS serial parameter by the driver. This property sets the time within which each single write operation must be completed at low level (Windows API) in the serial port.

DSR Timeout

Sets time-out value in milliseconds which will also be notified to the DSR serial parameter by the driver. This property sets the maximum time-out between the reception of one character and the next, which will also be the maximum time within which each individual read operation must be completed at low level (Windows API) in the serial port.

Display Dialog

When enabled (=True), it allows the Supervisor to display a dialog window upon connecting in Bridging to let the local user cancel the connection by remote control.

Disconn.Delay

Sets the delay time in milliseconds from receiving the disconnection signal, to effectively activating the closure of the bridging connection.

Connection String

Sets the string received from the modem which determines the remote connection request. Upon receiving this string the driver will activate the bridging service request.

Disconnection String

Sets the string received from the modem which determines the request to disconnect the remote connection. When receiving the string, the driver will deactivate the bridging service and restore the driver's communication.

Init String

Sets the modem's initialization string (i.e. AT&FS0=1).

OK String

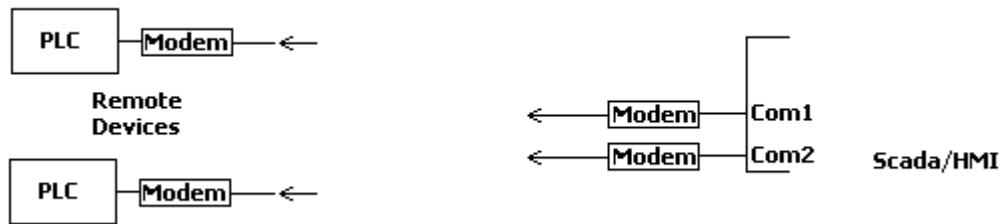
Sets the modem's OK string.

1.3.8. TAPI Settings (serial drivers only)

The TAPI functions allow the driver to connect to remote stations via modem in automatic and transparent mode.

When you need to connect remote PLCs or devices via modem to the Supervisor, the TAPI functions allow you to manage the communication protocol after having established the remote connections.

The diagram below demonstrates this type of connection:



The modem of the PLC Server must be configured ready to receive calls from the Supervisor. The Supervisor will automatically start a call to the destination modem, for the interested driver stations, when the tags belonging to those stations go in use.

The serial is managed by the modem's driver when the station's TAPI settings are used for managing communications via modem. The settings with which the serial is opened are therefore those defined in the modem's driver's advanced properties and may be different to those set for the Movicon driver station ("Serial Port Settings").



Please remember that the resources 'always in execution' (being Data Loggers, Alarms, Schedulers, General Logic) always keep the tags in use.



TAPI functions are enabled only by setting the 'Enable' property value to True

TAPI connections may behave in different ways when calls fail depending on the settings defined in the station. The combinations are as follows:

1. In situations using the station's status variable ("State/Command Variable") with the "Dial Only on Command" property enabled and calls fail, the following behaviour will result:
 - The "Bit 1" of the status variable is set at "True" after programmed call attempts have been made without success
 - An error will appear in the historical log describing the reason why the call was unsuccessful
 - The "Bit 1" of the status variable is automatically set at "False" by the driver when another connection is in command ("Bit 5")
2. In situations using the station's status variable ("State/Command Variable") with the "Dial Only on Command" property disabled and calls fail, the following behaviour will result:
 - The modem will start the call when the station's variables go in use (ie. upon loading a page)
 - The "Bit 1" of the status variable is set at "True" after programmed call attempts have been made without
 - An error will appear in the historical log describing the reason why the was unsuccessful
 - If the "Bit 1" is set at "False" (the station will reactivate), the driver will then try to call the station again if there are still active tasks towards the station

3. In situations without using the station's status variable ("State/Command Variable") and the call fails, the following behaviour will result:
 - The driver will try to connect to the station if there are still active tasks towards that station. Following attempts are those specified by the "Retries" parameter, then the connection goes on hold between one group of attempts and the next for the amount of time specified in the "Retry Hold Time (sec)" parameter

Phone Number

Sets the telephone number of the remote station to be connected to.

You can enter the name of a Supervisor string variable for drivers using driver base library build 250 and later. In this way you can define the phone number to call at runtime, assigning the proper value to this variable during supervision execution.

Retries

Defines the number of consecutive call attempts (without hold) in cases of connection failure to remote station. When set at the value "0", only one call attempt will be made, after which it will be put on hold for the time set in the "Retry Hold Time (sec)" parameter and then only one more call attempt will be made.

Disconnect After

Sets the time of inactivity, in seconds, before disconnecting. The connection is made as soon as the interested tags go into use in the project. When the interested tags are not in use, the Supervisor will disconnect after the set time expires.

Retry Hold Time (sec)

Defines the hold time in seconds between one group of call attempts and the next. The number of calls in each group is determined by the value set in the "Retries" + 1 parameter. There are no maximum limits on the number of call attempts which can be made. The driver will continue retrying for the length of time the station remains active and the driver's associated variables will continue to remain in use.

Enable

Enables, if set to True, the TAPI functions and calls via modem to the remote device.

Prompt before connect

When enabled (True), the system displays a dialog window asking confirmation before sending the call and activating the remote connection every time it has to execute connections via modem.

Show Dlg

When enabled, the system will display a dialog window to inform the user about the connection in action and its status.

Dial only on command (only available for drivers using driver base library build 250 and later)

When this property is enabled, the modem connection will be activated only on command, using the appropriate bit of the State/Command Variable (bit 5, Open modem connection), and stopped using the same variable (bit 6, Close modem connection). See "State/Command Variable" for details. This option is useful when communicating with multiple remote stations using only one modem to call. The designer can configure a driver station for each remote station, each one with a different phone number and a different State/Command Variable. At this point the user can connect to one remote station at a time using the specific State/Command variable. When data exchange with the current remote station is over, it will stop communications and then connect to another station.

1.3.9. TCP/IP Settings (ethernet drivers only)

In this group of settings you will need to set, for the selected station, the configurations inherent in the TCP-IP Ethernet access parameters.

Server Address

Specify the IP address or the name of the server or the network device to be connected to.

Examples: 192.168.0.1; localhost; server1

Server Port

The number of the TCP port of the server or device to be connected to. This value completes the device's IP address. For instance, the 502 port is always used for the TCP-IP Modbus (as established by the protocol), but when dealing with other devices please refer to their documentation.

Backup Server Address

The backup server IP address or name. If this address is set, the driver will try to connect to the backup server when unable to communicate with the 'primary' server. This happens in a redundancy situation at driver level. If communication with the backup server is interrupted, the driver will try to connect to the primary server again and so forth.

This field can contain the name of one Server or a list of Servers separated by the ";" character. The Server list allows more than one Backup Server for the Driver to manage in cyclic mode so that if one Server should fail the Driver can use the next on the list. When the Driver runs out of backup servers to use, it will start again with the Main Server.

Switch Server Timeout

The time entered here, in milliseconds, is the time which passes between a communication error verified on one server and an attempt to connect to the other one.

Local Bound Address

The local IP address is entered in this field, being the one from the PC's ethernet card you intend to use for communicating. Normally this property is left empty and used only when more than one ethernet card has been installed on the PC. The system will use the operating system's default address, when left empty.

Local Bound Port

The local TCP port address, referring to the PC's ethernet card which you intend to use for communicating, is entered in this field. Usually this field is left empty, unless required by the protocol or device being used.

Leaving the default value, the operating system will decide which port to use.

Use Ping

Enabling this option, the driver will use a ICMP protocol "ping" message to test for the existence of a Backup Server before attempting to connect to it. This permits faster connection to the Backup Server in cases where lists of Servers are very long. The timeout for responding to the "ping" message corresponds to the value set in the station's "Timeouts\Rx Timeout" field.

1.3.10. RAS Settings (ethernet drivers only)

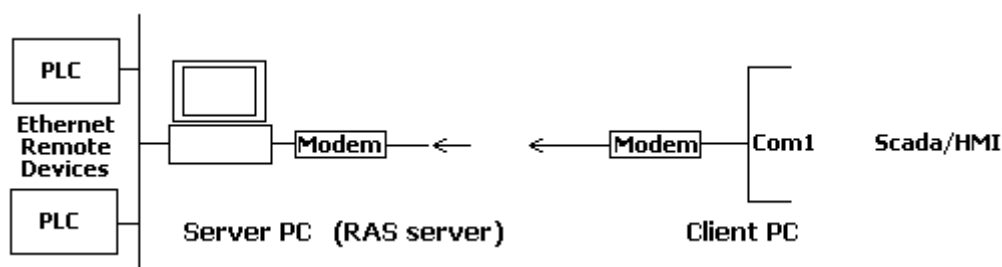
The RAS (Remote Access Service) connection is a Operating System function that allows a Server-Client type connection between two stations using a connection via Modem. Once the connection has been established the TCP/IP protocol can be used for exchanging data between Server and Client. The RAS connection can be created from the "Control Panel - Network Connections". The configuration procedure may change slightly according to the Operating System being used.

To make this possible you need to create an "Incoming Connection" on the PC Server, so that when the Client makes a call and connection is established, the two PCs will be linked as if they were networking with each other.

The driver's RAS functions allow you to connect the supervision to a remote network via modem in automatic and transparent mode.

When needing to connect to network devices from remote PCs, the RAS function lets you manage the network communication protocol after having established the connection with the modem through accessing a RAS Server, being a PC for accessing the network which can be reached via modem. After gaining RAS connection, the driver can access to the network.

The diagram below demonstrates this type of connection:



The modem of the RAS Server must be configured ready to receive calls from the Supervisor. The Supervisor will automatically start a call to the destination modem, for the interested driver stations, when the tags belonging to these stations go in use.

In this group of settings you need to set, for the selected station, the configuration inherent in the access parameters to remote devices by means of the RAS functions of the operating system.



Please remember that the resources 'always in execution' (being DataLoggers, Alarms, Schedulers, General Logic) always keep tags in use.



RAS functions are enabled by setting the 'Enable' property value to True only

Dial-up

Permits you to specify the name of one of the RAS Connections set in the operating system. When left empty, the driver will ask you to enter manually the telephone number, username and password of the station to be connected to when you try to connect.

The connection will be requested as soon as one of the associated tags goes into use in the running project.

If no predefined RAS connection has been specified in the Dial-up property, you need to specify a phone number, a user name and a password to access remote stations.

Phone Number

Sets the telephone number of the remote station to be connected to.

For drivers using driver base library build 250 and later you can specify the name of a Supervisor string variable here. By doing this you can define the phone number to call at runtime, assigning the right value to this variable during supervision execution.

User Name

Sets the username for accessing the remote station.

Password

Sets the password for accessing the remote station.

Retries

Sets the maximum number of connection attempts if the first one fails. When this number runs out and all attempts have failed, an error will be alerted.

Disconnect After

Sets the time of inactivity, in seconds, before disconnecting. The connection is made as soon as the interested tags go in use in the project. When the interested tags are not in use, the Supervisor disconnects after the set time expires.

Retry Hold Time (sec)

Sets the time, in seconds, the driver should wait before retrying to connect.

Enable

If set to True, the RAS functions and calls via modem to the remote device are enabled.

Prompt before connect

When enabled (True), the system displays a dialog window asking confirmation before sending the call and activating the remote connection every time it has to execute connections via modem.

Show Dlg

When enabled, the system will display a dialog window to inform the user about the connection in action and its status.

Dial only on command (only available for drivers using driver base library build 250 and later)

When enabling this property, the modem connection will be activated only on command, using the appropriate bit of the State/Command Variable (bit 5, Open modem connection), and stopped using the same variable (bit 6, Close modem connection). See State/Command Variable for details. This option is useful in case of communication with multiple remote stations using only one modem to call. The designer can configure a driver station for each remote station, each one having a different phone number and a different State/Command Variable. At this point the user can connect to one remote station at a time using the specific State/Command Variable. When data exchange with the current remote station is over, the operator can stop communication then connect to a different station.

1.3.11. Special TAPI and RAS configurations

RAS and TAPI techniques are used by the driver to manage supervision projects involving remote control. A number of different system architectures require special care in project design.

Most common remote control architectures are:

1. One remote station and a supervision using only one modem for communication
2. Various remote stations, placed in different areas, and a supervision using only one modem for communication
3. Various remote stations, located in the same area and connected in a network, and a supervision using only one modem for communication

Possible solutions for best communication management are:

1. No special configuration is needed in this case. One station should be defined for the driver and the call management can be both automatic (the driver starts a call when the tags go in use) or manual, using the State/Command variable for that station
2. Two or more stations should be defined, each one having its State/Command variable. The calls to different stations should be managed through the State/Command variable in order to avoid simultaneous calls to different stations
3. Two or more stations should be defined, each one having its State/Command variable, but only one station can be used for the RAS or TAPI call. Once the connection has been established for the "main" station, communication with other stations will be activated using the bit 1 of the State/Command variable for each station

For further detail about Station settings and State/Command variable see General (Station)

1.4. Task Settings

1.4.1. Tasks

In this selection you need to insert and set the static communication tasks when you intend to use this communication technique.



Please remember that the Supervisor offers you the possibility to set the driver's communication using the 'task' concept or the **dynamic tasks** concept. The dynamic tasks are automatically created by the driver at the project startup, based on the links to device's addresses set in the 'Dynamic Address' properties of each single tag.

The communication tasks allow '**static**' tasks to be assigned to the driver, which will be executed polling the device provided.



By using the communication technique in tasks, you need to set, in static mode, the relationship between the Supervisor variables and the device addresses.

Add

The 'Add' button allows you to insert a new static Task for the communication driver. When a new task is inserted, a window will automatically be displayed to set the requested parameters. The inserted tasks can be edited or removed afterwards by using the buttons described below.

Edit

The 'Edit' button allows you to change the parameters of existing tasks. You first need to select the task desired and then use the 'Edit' button or double-click.

Remove

The "Remove" button allows you to delete existing tasks. You first need to select the task desired and then use the 'Remove' button.

1.4.2. Static Tasks

Station

The name which identifies the station corresponds to the device you intend to communicate with. At least one station must be set for each task.

When more than one station has been defined, you need to select here the station where the task is to be executed.

Task Name

The name which identifies the task. Any identification string can be used to identify the Task. Each task must have a unique name.

Variables

Allows you to associate which Supervisor's tag (or tags list) the task is to manage. Using the ellipse button to the right you can select any tag previously inserted into the project's Real Time Database or create new tags. These will be added automatically in the project's Real Time Database. The tags in Variables field must be **separated by a ';' character** and **should be consecutive starting from the device address set in Device address field**.



It is strongly recommended that the variables list includes tags of the same data type, otherwise unpredictable errors may occur.

A structure tag, whose members have different data types can be used to address a group of tags, with different data types, with a static task.

The driver automatically calculates the number of bytes to read/write from/to the device starting from the device address by adding the byte size of each tag present in the 'Variables' Task Property.

The driver will then read or write a byte-aligned, raw buffer whose starting address is the one specified in the 'Device Address' property, and map it to each tag, according to each tag's offset and size in the buffer.

The driver performs additional checks according to the specified addressing:

Bit Addressing

When accessing bits on the device, all tags in the 'Variables' list must be declared in the Tag Database as Bit type, otherwise an error will show. The number of bits to be exchanged with the device equals the number of tags in the 'Variables' list. The starting address is the one specified in the 'Device Address' property.

Byte, word or double word addressing:

None of the tags in the 'Variables' list must be declared in the Tag Database as Bit type, otherwise an error will show.

Conditional Variable

Allows you to associate a project variable whose status will determine the communication task's execution condition. By using the ellipse button to the right you will be able to select any tag previously inserted into the project's Real Time Database or create new ones. These will be added automatically in the project's Real Time Database.

The variable (of any type) will therefore condition the task's execution: when set to a value other than zero (>0) the communication task will be executed by the driver.



When the execution of the task has been completed the driver automatically sets the value of the conditional variable to zero. Therefore this needs to be taken into account when variables are managed by logic.

Polling Time

This parameter, expressed in milliseconds, determines the minimum polling time of the each single task's execution for updating data **when the variables are in use**.

The Polling Time default value, inserted when creating the task, is determined by the value established in the driver's General properties ("Polling Time"), but can be modified as pleased for each single task. The value equal to zero means that data will get updated at the highest possible speed.

A higher value can be set, for instance, when data does not require fast updating times.

Unused Polling Time

This parameter, expressed in milliseconds, allows you to force a data update of each task even **when the variables are not in use**, by establishing a polling time.

The Unused Polling Time default value, inserted when creating tasks, is determined by the value established in the driver's General properties ("Unused Polling Time"), but can be changed for each single task as pleased.

Setting this value to 10000 (being 10 seconds) means that the task will be executed with the minimum time of 10 seconds even when its variables are in use.



When this parameter is set = 0, the tasks will not be executed when the variables are not in use.

When this parameter is higher than 0, ALL the project tags that are 'not in use' are read with this frequency, if the task is a read or read/write one. This may slow the supervision down if many tags are defined in project.

Address offset variable (only available for drivers using driver base library build 250 and later)

Specify here a tag name of numeric integer type. The tag's value can be modified at runtime and will be used as an offset +/- with respect to the starting address set for the task. It will be possible to read in different points of the device memory area (before or after the start address) by changing the address offset tag value at runtime.

The offset unit depends on the driver and on the address type:

- for addressing method based on bit, an offset value = 1 corresponds to one bit
- for drivers whose addressing method is based on bytes, an offset value = 1 corresponds to one byte
- for drivers whose addressing method is based on words, an offset value = 1 corresponds to one word

The user who sets the value for the address offset tag should use meaningful values: i.e. to read the next word using a Siemens driver the offset value should be set to +2, since the valid addresses for word tags are only the even values. In cases where the Modbus driver is concerned, the offset value should be set to +1 since its addressing is based on words.

Caution: also when changing the offset value, the tasks will not be executed immediately but only when needed. Therefore, for instance, an Exception output task will be executed with a new offset only when the associated variable's value is modified. So you can check whether a task has been executed with the conditional variable.

As an example, in order to write the same value in more addresses using the offset tag, you can create an Unconditional Output task with a conditional variable. The write operation will be executed changing the value of the offset tag and setting the conditional variable. When reset, the task has been completed and the offset tag's value can be change to write the following address.

Swap Byte

This selection allows you to swap the bytes in 'word' type data. When doing this the data linked to the Supervisor and the device will have the 'high' byte swapped over with the 'low' byte and vice-versa, for each word type data of the task.

Swap Word

This selection allows you to swap the words in 'double words' type data. By doing this the data linked to the Supervisor and the device will have the 'high' word swapped over with the 'low' word and vice-versa, for each double word type data of the task.

Type

In this selection you can set the execution type you want to assign to the task.

The options offered are:

| | |
|-----------------------------|--|
| Input | This option sets the task as 'Read Only'. So when the variables are in use, the driver executes read polling operations in the device and transfer the read data values to the related project variables. |
| Input/Output | This option sets the task as 'Read-Write'. So when the variables are in use, the driver will execute the read polling operations in the device, and transfer read data values to the related project variables. When a variable changes its value in the Supervisor, the driver writes data to the device, then goes back to reading. |
| Exception Output | This option sets the task as 'Write Only', thus managing data on exception, which means only when there is a data change in the Supervisor. |
| Unconditional Output | This option sets the task as 'Write Only', thus continuously writing to the device, independently of data changes. |

Write Outputs at Startup

This property is meaning card you need to define the settings inherent in the 'General' properties group for the selected task.

Device Data

The parameters of this group refer to addressing the memory areas of the specified device.
Please refer to the documentation of the specific driver and to protocol specifications.

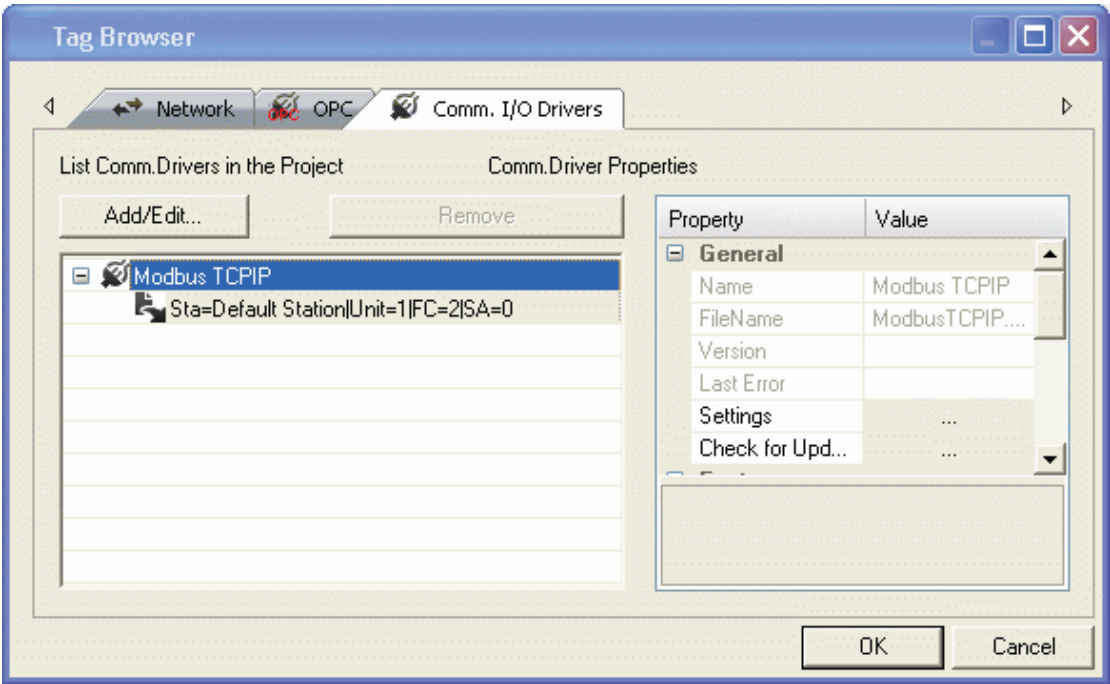
1.4.3. Dynamic Tasks

Dynamic tasks are automatically created by the driver at project startup, based on the links to device's addresses set in the 'Dynamic Address' properties of each single tag.
Settings for dynamic tasks are not configured through the "Driver Settings " window but are specified for each tag through the "Tag Browser" window.

To set 'Dynamic' properties for a tag, select the 'Real Time DB' resource and the 'List Variables' node in the tree from the Supervisor 'Project Explorer' window. Select the tag of interest, then go to the 'Properties' window and click on the ellipse button to the right of 'Dynamic' property.
The 'Tag Browser' dialog will open, choose the 'Comm Drivers' tab and double click on the most suitable driver to get the 'Task properties' page.

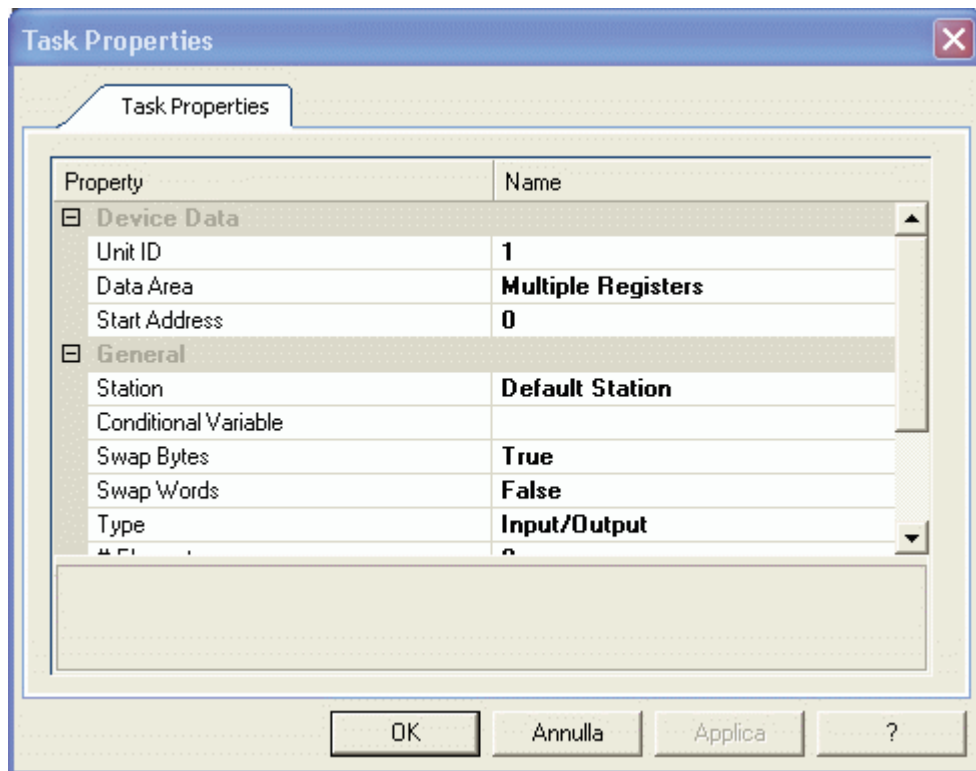
Inserting Tag "Physical I/O addresses"

To set the dynamic properties of tags, select the RealTimeDB list variables (tag) from the Project Explorer window, then select the individual tag desired and display its Properties Window. Click the button to the right of the "Dynamic" property to open the "Tag Browser" window.



Tag Browser Window.

Select the "Comm. I/O Drivers" tab to view the list of drivers, inserted in the project, with their predefined dynamic links. You will also find an "Add/Edit..." button, used for opening the "Task Properties" window for editing the dynamic links, and a "remove" button used for deleting the pre-selected dynamic link.



Task Properties window.

Once the "Tag Browser" window has been opened, you can insert/edit dynamic links in the following ways:

Inserting a new dynamic link

1. open the "Tag Browser" window through the "Dynamic" property
2. select the name of the driver you wish to use
3. click on the "Add/Edit..." button to open the "Task Properties" window
4. if the variable does not have any assigned dynamic links, the "Task Properties" window will open showing its default values. On the other hand, if the variable has an assigned dynamic link, the "Task Properties" will open showing the value of that link. Set the parameters desired
5. close the "Task Properties" with "OK". Upon doing this the dynamic address will be composed with the set values and added to the new link in the list of dynamic addresses and focused on
6. close the "Tag Browser" window with Ok to insert the dynamic link into the variable's "Dynamic" property

Editing an existing dynamic link

1. open the "Tag Browser" window from the variable's "Dynamic" property
2. select the desired dynamic link from driver list and click on the "Add/Edit..." button to open the "Task Properties" window
3. if the variable does not have any assigned dynamic links, the "Task Properties" window will open showing the values of the selected dynamic link. If the variable already has a dynamic link, the "Task Properties" will open with the values of that link. Therefore set the parameters desired
4. close the "Task Properties" window with "OK" upon which the dynamic address will be composed with the set values updating the link in the list of dynamic addresses and given focus
5. close the "Tag Browser" window with Ok to insert the dynamic link into the variable's "Dynamic" property

Selecting an existing dynamic link

1. open the "Tag Browser" window from the variable's "Dynamic" property
2. select the desired dynamic link from driver list
3. close the "Tag Browser" window with Ok to insert the dynamic link into the variable's "Dynamic" property

Keep in mind that the list of a driver's dynamic links displayed in the Tag Browser window is only used by the programmer as a reminder or for selecting. Once a link has been assigned to the variable it can be removed from the list without causing any damage to the project. The dynamic link list is saved in the ".dyndrv" file of the driver in the project's "RESOURCES" folder. If you delete this file, the list of dynamic links inserted in the "Tag Browser" window up to that moment will no longer be

available. Only the dynamic address previously assigned in the variables' "Dynamic" properties will remain saved.

Dynamic Task Properties

The dynamic link settings of variables can be defined using the "Task Properties" window. In this window you can assign the task's general properties in the 'General' group, and the those properties used for entering the device addresses which vary according to device type. Below you will find a description on each of the task's general properties. Please refer to the Driver's help for device address specifications.

Station

The name which identifies the station corresponds to the device you intend to communicate with. At least one station must be set for each task.

When more than one station has been defined, you will need to select the station where the task is to be executed.

Conditional Variable

Allows you to associate a project variable whose status will determine the communication task's execution condition. Using the ellipse button to the right you can select any tag previously inserted into the project's Real Time Database or create new tags which will then be added automatically in the project's Real Time Database.

The variable (of any type) will therefore condition the task's execution: when set to a value other than zero (>0) the communication task will be executed by the driver.

When the execution of the task has been completed the driver will automatically set the value of the conditional variable to zero. Therefore this needs to be taken into account when variables are managed by logic.

Swap Byte

This selection swaps bytes over in "word" type data. This means that data linked between the Supervisor and device will be seen with the "high" byte inverted with the "low" byte and vice versa, for each task word data type.

This property is only available for some drivers.

Swap Word

This selection swaps words over in "double word" data types. This means that data linked between Supervisor and device will be seen with the "high" word swapped over with the "low" word and vice versa for each task double word data type.

This property is only available for some drivers.

Type

In this selection you can set the execution type you want to assign to the task. Possible values are:

| | |
|-----------------------------|--|
| Input | This option sets the task as 'Read Only'. So when the variables are in use, the driver executes read polling operations in the device and transfer the read data values to the related project variables. |
| Input/Output | This option sets the task as 'Read-Write'. So when the variables are in use, the driver will execute the read polling operations in the device, and transfer read data values to the related project variables. When a variable changes its value in the Supervisor, the driver writes data to the device, then goes back to reading. |
| Exception Output | This option sets the task as 'Write Only', thus managing data on exception, which means only when there is a data change in the Supervisor. |
| Unconditional Output | This option sets the task as 'Write Only', thus continuously writing to the device, independently of data changes. |

#Elements

Specifies the number of elements to be read/written by the driver. Possible values are 0 and 1 only. If left at zero, the driver automatically calculates the number of elements to be read/written in the device data area, on the basis of the tag data type. For example, in a WORD area, for a tag whose data type is word one element is read/written, whereas two elements are read/written for a float tag. If set to one, only one element is read/written, no matter what the tag data type is. For example, in a WORD area, only one element (one word) is read/written for a float tag.

Write Outputs at Startup

This property is meaningful only for Input/Output or Exception Output tasks. When set to True the task is executed in output when the project starts up.

1.5. Import Device Database

1.5.1. Import Device Database

Thanks to this important feature, you can directly access the database of a PLC or an equivalent data source of a device to import the desired Tags into the Supervisor project.

When terminating this operation, the project's Real Time Database will automatically fill up, by inserting all the imported tags, which will be defined as Dynamic Tags in the 'Not Shared' areas, with the corresponding tag type and with the address already assigned for the device.

When right clicking on the Driver name in the Project explorer window and selecting "Import Device Database", you will be prompted to select the 'data source', being the PLC database, the symbolic file or the .CSV file obtained from the PLC data or device explorer.

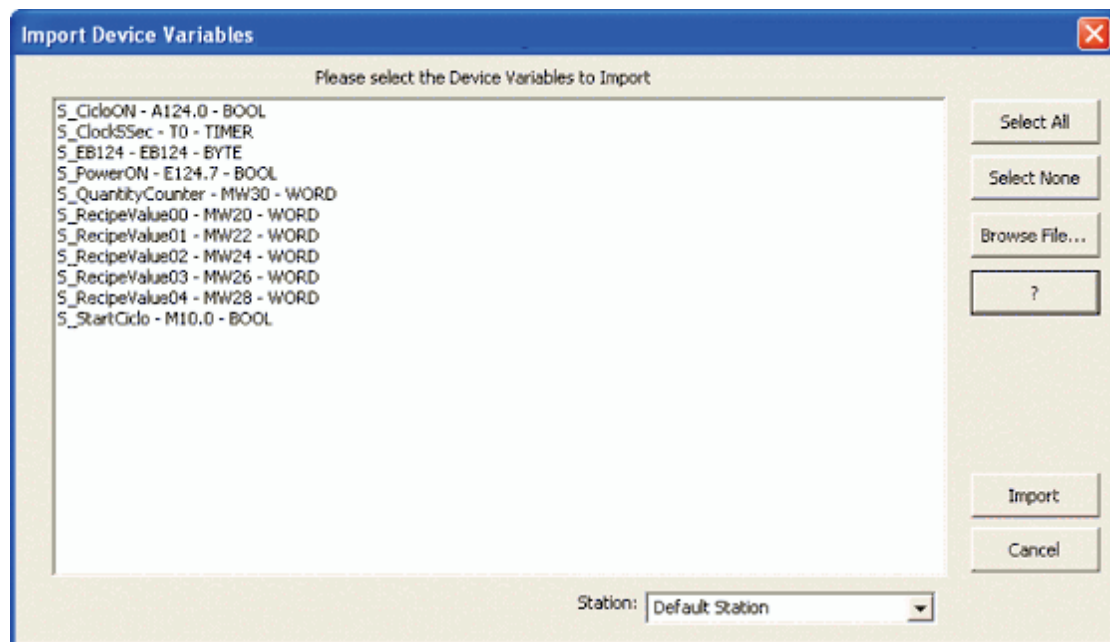
After getting the file desired, you will be shown the window, as illustrated below, which displays the variables it contains.



Attention: Importing data from the PLC is supported in all the drivers for the most well-known devices. Check the access modalities or 'data source' requirements, which may vary from one device to another.



If variables that are already in the RealTimeDB are recovered while importing data, they will be overwritten without requesting confirmation beforehand. Those that will be overwritten include the "Data Type" and "Fixed I/O Address" properties but the "Tag Description" property will remain the same and will only be imported if the variable is not already in project's RealTimeDB.



The import operation will take longer to complete when a large number of variables has been selected and can however be aborted by using the "ESC" key.

Importing Structures and Arrays using the device's Data Base's import tool can be done in two different modes:

If the Structure's root is selected from the tool's dialog window following by activating the "import" button, a prototype defined with the structure's members and a variable of the same prototype will be created in the project. After this happens, it will then be possible to import a multiple selection of structures. This can also be done using the "Select All" button and then the "Import" button.

If one or more members of one or more Structures are selected and then imported, variable types corresponding to those of the selected members will be created and not those of the structure prototypes. The same result can be obtained clicking on the "Expand All" button to explore the Structures' members, then clicking on the "Select All" button to select them all and then finally on the "Import" button to import each member as distinct variables.

If each single variable as well as the Structure variables are on the device's variable list, when used the "Select All" button will select them all.

When importing Arrays, select then import the root of the Array to create an Array type variable with its "Element Type for Array" property set to the Array type. The same can be done using the "Select All" button and then the "Import" button.

If one or more elements or Arrays are selected and then imported, variables corresponding to the selected elements will be created but not the Array type variables. This same result can be obtained by clicking on the "Expand All" button to explore the Array elements, then clicking on the "Select All" for selecting them all, followed by clicking the "import" button to import each single element as distinct variables.

If each single variable as well as the Array variable are on the device's variable list, the "Select All" button will select them all when clicked on.

Select All

Allows you to select all the variables from the importation file.

Use the CTRL+Click or SHIFT+Click keys for partially selecting variables.

Select None

Allows you to deselect all the variables from the importation file.

Use the CTRL+Click or SHIFT+Click combo keys for partially deselecting variables.

Browse File...

Allows you to change the origin file, by activating the standard window for file selection.

? (help)

Activates the guide containing information relating to the origin data format requirements.

Import

Activates the importation of variables from the origin file (device's data source) to the Supervisor project. When the importation has terminated, the project's Real Time Database resource will result as being populated with all the imported variables.



As the 'data sources' depend on the device and might change, it is advised to always check the imported variables' properties to see whether the automatic parsing, the type assigned and the device's address have been executed correctly when the importation has terminated.

Cancel

This button cancels the importing operation.

Station:

This box allows you to select the driver's station to be assigned to the imported variables, when the driver has been set with more than one station.

1.6. API Basic Script Interfaces

1.6.1. Basic Script Interface Use

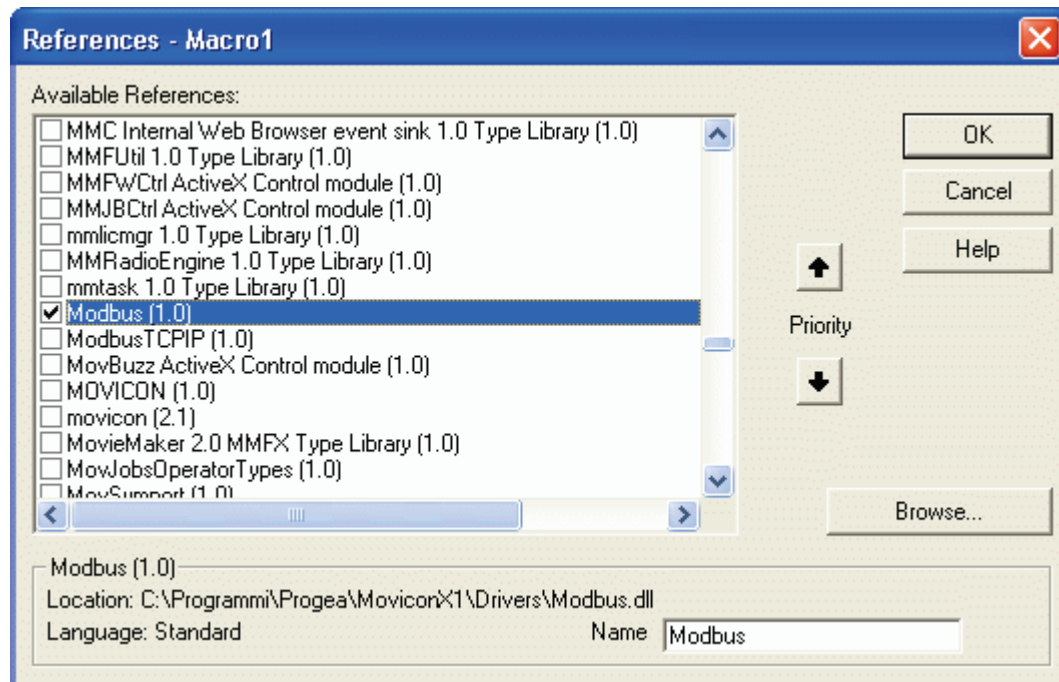
The Supervisor allows you to use a set of functions for managing communication drivers with Basic Scripts as well. In this way you can, for example create and destroy communication tasks during runtime, or retrieve information on the status of station tasks, etc.



In order to use the COM interface (Component Object Model) of a communication driver, you will need to enable the "VBA Driver" option on the license. Otherwise the driver's basic script functions will not get executed in runtime mode.

To make programming easier you can enable the interested Driver Reference within the basic script routine to declare the specific necessary object types (ie. DriverInterface, TaskInterface, etc.) and

not generic types (Object). This will give you a list of methods and properties of each specific object you declare.



Please keep in mind that the Reference enabling is not supported on the WinCE platform. If you have to develop a project which has to be executed on WinCE, the References mustn't be enabled but you must declare all Object types.

The examples shown in function documentation have all been created by enabling the driver's Reference.

The following example can only work on Windows 32/64 bit platforms.

Sub Main

```
Dim drv As DriverInterface
Dim station As StationInterface
Dim bError As Boolean
```

```
Set drv = GetDriverInterface("Modbus Serial")
Set station = drv.GetDriverStation("Station1")
```

```
bError = station.IsInError
MsgBox "IsInError = " & CStr(bError),vbInformation,"ERROR"
```

```
Set drv = Nothing
Set station = Nothing
```

End Sub

To make it compatible with the WinCE platform the driver References must be enabled and the variables should be declared as follows:

Sub Main

```
Dim drv As Object
Dim station As Object
Dim bError As Boolean
```

```
Set drv = GetDriverInterface("Modbus Serial")
Set station = drv.GetDriverStation("Station1")
```

```
bError = station.IsInError
MsgBox "IsInError = " & CStr(bError),vbInformation,"ERROR"
```

```
Set drv = Nothing
Set station = Nothing
```

End Sub

Synchronous Tasks and Asynchronous Tasks

When a task is executed using the "Execute()" function you can choose whether the task should be executed in Sync. mode or Async. mode:

Synchronous Tasks

To make a task become synchronous just assign a timeout time higher than zero to the "Execute()" function. Therefore when this function is called, the script will not be executed until the function has terminated. If the function is executed successfully it will return the "True" value. Otherwise it will return the "False" value if not executed, which may mean that timeout expired before completion.



The "Execute()" function returns the "True" value when it has been executed, meaning that the task was executed but there may be an error and therefore you will have to verify this by controlling the Station's status.

Synchronous Tasks are usually used in cases where data read/writes need to be controlled. For example, to double-check whether a production recipe has been transferred before giving start production command.



To make sure the synchronous task works correctly you will need to set the "Execute()" function's timeout parameters with a higher value to that set in the Station's timeout. For instance, if the station's timeout is 2000 msec, you can then set the "Execute()" function's timeout with 5000 msec. This will ensure that the function's timeout does not expire before the Station's timeout does.

Asynchronous Tasks

To make a task asynchronous, just assign the "Execute()" function with timeout equal to zero. Therefore when the function is called, the script's execution will continue without waiting for the function to terminate. The task will therefore be executed when possible but it will not be possible to know exactly when it happened. In this case the function will always return a "True" value if no error has occurred in the task's creation, such as a wrong parameter. The "True" value for an asynchronous task does not necessarily mean that it has been executed but has just simply been created and is waiting to be executed.

1.6.2. DriverInterface

DriverInterface Events

FireOnDataSent, DriverInterface Event

| | |
|--------------------|--|
| Description | This event is fired when data has been sent by a task. It can be used to retrieve new information without polling. |
| Remarks | Only available from external COM clients. Not available from Supervisor BasicScripts. |

| Parameter | Description |
|--------------------------------|---|
| (ByVal IpszTaskName As String) | The name of the task in which data has been sent. |

Example:

FireOnNewData, DriverInterface Event

| | |
|--------------------|--|
| Description | This event is fired when data has changed on a task. It can be used to retrieve new information without polling. |
| Remarks | Only available from external COM clients. Not available from Supervisor BasicScripts. |

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

| | | |
|-------------------|-----------------|---|
| (ByVal String) | IpszTaskName As | The name of the task in which data has changed. |
|-------------------|-----------------|---|

Example:

```
Option Explicit
Dim WithEvents Drv As ModbusTCP/IP.DriverInterface
Dim task As ModbusTCP/IP.TaskInterface
Dim task2 As ModbusTCP/IP.TaskInterface
Dim var As Variant

Private Sub Drv_FireOnNewData(ByVal IpszTaskName As String)
    If IpszTaskName = "Dyn1" Then
        var = task.GetReadByteBuffer
        Label1.Caption = CStr(var)
    ElseIf IpszTaskName = "Dyn2" Then
        var = task2.GetReadByteBuffer
        Text2.Text = CStr(var)
    End If
End Sub
```

FireOnQualityChanged, DriverInterface Event

- Description** This event is fired when data quality has changed on a task. It can be used to retrieve new information without polling.
- Remarks** Only available from external COM clients. Not available from Supervisor BasicScripts.

| Parameter | Description |
|-------------------|--|
| (ByVal String) | IpszTaskName As The name of the task in which data has changed quality. |

Example:

```
Dim WithEvents Drv As ModbusTCP/IP.DriverInterface
Dim task As ModbusTCP/IP.TaskInterface
Dim var As Variant

Private Sub Drv_FireOnQualityChanged(ByVal IpszTaskName As String)
    Dim Station As ModbusTCP/IP.StationInterface
    Set Station = task.GetStationObject
    lblQuality = CStr(Station.Quality)
    If Station.IsInError Then
        lblError = Station.LastErrorString
    Else
        lblError = "No Error"
    End If
End Sub
```

DriverInterface Functions

AddDriverTask, DriverInterface Function

- Syntax** AddDriverTask(IpszName, IpszSettings)
- Description** Adds a Dynamic Task to a driver. A new Dynamic Task is created and its resources are allocated, but it will not be executed until a call to the Execute function is made. The task will remain allocated in the driver until removed by a call to RemoveDriverTask.
- Remarks**

| Parameter | Description |
|--------------------|----------------------------|
| IpszName As String | Task Name. Must be unique. |

| | | |
|---------------------|----|---|
| lpszSettings String | As | <p>String containing the Dynamic Settings for the task. Please refer to each driver's specific documentation for Dynamic Settings formats. This string is the same one inserted in the "Dynamic" variable's property.</p> <p>As a general rule, Dynamic Settings strings are a list of keywords separated by the " " (pipe) character. For instance, the Modbus driver may have:</p> <p>[DRV]Modbus TCPIP.Sta=ModbusTCP Typ=10 Size=10 TaskType=0 Unit=1 FC=2 SA=20</p> <p>where:</p> <p>[DRV]: this is the parameter that indicates driver use and must be followed by driver's name, in the example this will be "Modbus TCPIP"</p> <p>Sta: this is the parameter that indicated the name of the reference station in which task is created. In the example the station will be called "ModbusTCP"</p> <p>Typ: indicates data type to be exchanged. The following values may be used in this parameter:</p> <p>0 = Bit 1 = Sign Byte 2 = Byte 3 = Sign Word 4 = Word 5 = Sign DWord 6 = DWord 7 = Float 8 = Double 9 = String 10 = Array 11 = Structure</p> <p>Size:this is the parameter that indicates the data size in bytes to be exchanged by the task</p> <p>TaskType:this is the parameter that indicates type of task to be created. The following values may be used in this parameter:</p> <p>0 = Task Input 1 = Task Input/Output 2 = Task Exception Output 3 = Task Unconditional Output</p> <p>Unit:this is the parameter that indicates the unit number. This parameter is only available for the Modbus driver only</p> <p>FC: this is the parameter that indicated the function code to use. This parameter is only available for the Modbus driver only</p> <p>SA: this is the start address parameter that indicates the start address from where data is to be read/written on device.</p> |
|---------------------|----|---|

Result TaskInterface

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim Station As StationInterface
    Dim TaskReset As TaskInterface
    Dim sDynString As String
    Dim vBuffer As Variant

    'Create driver object
    Set drv = GetDriverInterface("Modbus TCPIP")
    If drv Is Nothing Then
        MsgBox "Modbus TCPIP driver not installed in the project!",
            vbExclamation, GetProjectTitle
        Exit Sub
    End If

    'Create station object

```

```

Set Station = drv.GetDriverStation("Station1")
If Station Is Nothing Then
    MsgBox "Station1 didn't found in the driver stations list!", vbExclamation,
    GetProjectTitle
    Exit Sub
End If

If Station.IsInError Then
    MsgBox "Station in Error!", vbExclamation, GetProjectTitle
    Exit Sub
End If

sDynString = "[DRV]Modbus
TCPIP.Sta=Station1|Typ=4|Size=2|TaskType=2|Unit=1|FC=2|SA=100"

'Create task object
Set TaskReset = drv.AddDriverTask("Dyn1", sDynString)
If TaskReset Is Nothing Then
    MsgBox "the dynamic string '" & sDynString & "' is wrong!",
    vbExclamation, GetProjectTitle
    Exit Sub
End If

If Not TaskReset.IsValid Then
    MsgBox "AddDriverTask Failed!", vbExclamation, GetProjectTitle
    Exit Sub
End If

TaskReset.PollingTime = 0

Dim ret As Boolean

'Reset buffer value
vBuffer = 0

ret = TaskReset.SetWriteByteBuffer(vBuffer)

If ret = False Then
    MsgBox "SetWriteByteBuffer Failed!", vbExclamation, GetProjectTitle
    Exit Sub
End If

'Execute a sync write command
ret = TaskReset.Execute(5000)

If ret = False Then
    MsgBox "TaskReset.Execute Failed!", vbExclamation, GetProjectTitle
    Exit Sub
End If

'Remove task
ret = drv.RemoveDriverTask("Dyn1")

If ret = False Then
    MsgBox "RemoveDriverTask Failed!", vbExclamation, GetProjectTitle
    Exit Sub
End If

Set Station = Nothing
Set drv = Nothing
Set TaskReset = Nothing
End Sub

```

GetDriverStation, DriverInterface Function

Syntax GetDriverStation(IpszStation)

Description Returns a Station object.

Remarks

| Parameter | Description |
|-----------------------|---------------|
| IpszStation As String | Station Name. |

Result Object
If Function has been executed successfully it will retrieve an object of type StationInterface if otherwise Nothing is returned.

Example:
See the "AddDriverTask" function example.

GetDriverTask, DriverInterface Function

Syntax GetDriverTask(IpszName)

Description Returns a task object. This function can be used to retrieve the interface to any static or dynamic task.

Remarks

| Parameter | Description |
|--------------------|-------------|
| IpszName As String | Task Name. |

Result Object
If Function has been executed successfully it will retrieve an object of TaskInterface type if otherwise Nothing is returned.

Example:

Sub Main

```
Dim drv As DriverInterface
Dim Station As StationInterface
Dim task As TaskInterface
Dim var As Variant
Dim nVal As Integer
Dim nVal2 As Integer
Dim AppByte As Byte

Set drv = GetDriverInterface("Modbus TCP/IP")
Set task = drv.GetDriverTask("AsyncW")

If task Is Nothing Then
    MsgBox "Async task not found"
    Exit Sub
End If

While Not This.IsStopping
    var = task.GetReadByteBuffer
    If (var(1) And &H80) > 0 Then
        AppByte = Not(var(0))
        nVal = AppByte + 1
        AppByte = Not(var(1))
        If nVal > 255 Then
            nVal2 = (((AppByte + 1) * 256) + (nVal And &HFF)) * -1
        Else
            nVal2 = ((AppByte * 256) + (nVal And &HFF)) * -1
        End If
        SetVariableValue("AsyncRead", nVal2)
    Else
        SetVariableValue("AsyncRead", var(0) + (var(1)*256))
    End If
DoEvents
```

```

Wend

Set Station = Nothing
Set drv = Nothing
Set task = Nothing
End Sub

```

InitDriver, DriverInterface Function

Syntax InitDriver(lpszSerialNumber, lpszFileSettings)

Description Initialises a Communication Driver. Once this function has been called successfully, the driver is ready to operate.

Remarks Only available from external COM clients. Not available from Supervisor BasicScripts

| Parameter | Description |
|----------------------------|---|
| lpszSerialNumber As String | Driver Serial Number. Contact Sails Office for a valid Serial Number for the Driver you want to use. |
| lpszFileSettings As String | Path to a Directory containing the Driver Settings File. The file name must have the following format: <DriverDLLName>.drvsettings Where "DriverDLLName" is the name of the Dynamic Link Library for the Driver. The file is in XML format and can be written in 3 ways: <ol style="list-style-type: none"> 1. Manually with a text editor 2. By a previous call to the SetDriver function 3. By using the Supervisor user interface (design mode) |

Result Boolean
True = the driver has been initialised successfully
False = driver initialization failed

Example:

```

Private Sub Form_Load()
    Dim bInitOk As Boolean
    Set Drv = CreateObject("SupervisorDriver.ModbusTCP/IP.1")
    'Drv.SetDriver "C:\Projects\TestPrj\RESOURCES\TestPrj\"
    bInitOk = Drv.InitDriver("XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "C:\Projects\TestPrj\RESOURCES\TestPrj\")
    If Not bInitOk Then
        MsgBox "InitDriver failed!", vbExclamation, "ERROR"
    End If
End Sub

```

RemoveDriverTask, DriverInterface Function

Syntax RemoveDriverTask(lpszName)

Description Removes a Dynamic Task from a driver. A Dynamic Task is removed and its resources are freed.

Remarks

| Parameter | Description |
|--------------------|-------------|
| lpszName As String | Task Name. |

Result Boolean
True = function has been executed successfully
False = function execution failed

Example:

See the "AddDriverTask" function example.

RemoveStation, DriverInterface Function

Syntax RemoveStation(lpszName)

Description Destroys and removes a station from the driver's station list. this function removes both stations dynamically added with AddStation and those created through the driver's user interface (file. DrvSettings).

Remarks This function stops the execution station's defined tasks. In addition, if executed in stations that already existed when starting the project up in runtime, this function will render the Station unusable in runtime but will not cancel its XML data from the .drvsettings file.

| Parameter | Description |
|--------------------|---------------|
| lpszName As String | Station Name. |

Result Boolean
If the function is successful it returns the value True.

Please note To permanently remove (in the file DrvSettings) call the function SaveConfigDriver.

Example:

```
Sub Main
    Option Explicit
    Const DRIVER_NAME = "Modbus Serial"

    Sub Main
        Dim objDrvInt As DriverInterface
        Dim sStationName As String
        Dim bRet As Boolean

        sStationName = This.GetParameter(0)

        If sStationName <> "" Then

            Set objDrvInt = GetDriverInterface(DRIVER_NAME)

            'REMOVE STATION
            bRet = objDrvInt.RemoveStation(sStationName)
            End If

            If Not bRet Then Debug.Print "VBA Driver - RemoveStation Error! Station
            name: " & sStationName & ""

            Set objDrvInt = Nothing

        End Sub
```

SetDriver, DriverInterface Function

Syntax SetDriver(lpszFileSettings)

Description Displays a Property Sheet dialog box, by which the user can change a driver's General and Station settings.

Remarks Only available from external COM clients. Not available from Supervisor BasicScripts.
This function must be called when the driver is not initialised, that is before InitDriver is called, otherwise it will fail.

| Parameter | Description |
|----------------------------|--|
| IpszFileSettings As String | Path to a Directory containing the Driver Settings File. The file name must have the following format: <DriverDLLName>.drvsettings Where "DriverDLLName" is the name of the Dynamic Link Library for the Driver. |

Result Boolean
True = the function has been executed successfully
False = function execution failed

Example:

```
Private Sub Form_Load()
    Dim bInitOk As Boolean
    Set Drv = CreateObject("SupervisorDriver.ModbusTCP/IP.1")

    Dim bRet As Boolean
    bRet = Drv.SetDriver("C:\Projects\TestPrj\RESOURCES\TestPrj\")
    If Not bRet Then
        MsgBox " SetDriver failed!", vbExclamation, "ERROR"
    End If
End Sub
```

TerminateDriver, DriverInterface Function

Syntax TerminateDriver()

Description Terminates a Communication Driver previously initialised by the InitDriver function.

Remarks Only available from external COM clients. Not available from Supervisor BasicScripts

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
True = the driver has been terminated successfully
False = driver termination failed

Example:

```
Private Sub Form_Load()
    Dim bInitOk As Boolean
    Set Drv = CreateObject("SupervisorDriver.ModbusTCP/IP.1")
    'Drv.SetDriver "C:\Projects\TestPrj\RESOURCES\TestPrj\"
    bInitOk = Drv.InitDriver("XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
        "C:\Projects\TestPrj\RESOURCES\TestPrj\")
    If Not bInitOk Then
        MsgBox "InitDriver failed"
    End If

    If not Drv.TerminateDriver()
        MsgBox "TerminateDriver failed!", vbExclamation, "ERROR"
    End If
End Sub
```

AddStation, DriverInterface Function

Syntax AddStation(IpszName)

Description Dynamically creates a new station, and inserts it into the list of stations Driver. When a station is created its resources are allocated, but to put running, you must use the "StartStation."

Remarks A station can have any name, as long as unique and not null.

| Parameter | Description |
|--------------------|---|
| IpszName As String | Station Name to be created. It must be a unique name. |

Result Boolean
If the function is successful it returns the value True. If the station has added a zero or station name exists in the station list is returned False.

Note: The new station is additionally configured with default property values.

Note: To permanently store the new station (in the file. DrvSettings) invoke the function "SaveConfigDriver".

Example:

```
Option Explicit
Const DRIVER_NAME = "Modbus Serial"

Sub Main
Dim objDrvInt As DriverInterface
Dim sStationName As String
Dim bRet As Boolean

sStationName = This.GetParameter(0)

If sStationName <> "" Then

Set objDrvInt = GetDriverInterface(DRIVER_NAME)

'ADD STATION
bRet = objDrvInt.AddStation(sStationName)
End If

If Not bRet Then Debug.Print "VBA Driver - AddStation Error! Station
name: " & sStationName & ""

Set objDrvInt = Nothing

End Sub
```

SaveConfigDriver, DriverInterface Function

Syntax SaveConfigDriver ()

Description SSave permanently (in the file. DrvSettings) the configuration parameters drivers. Can be used, for example, to save a station dynamically added with the function AddStation.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
If the function is successful it returns the value True.

Example:

```
Option Explicit
Const DRIVER_NAME = "Modbus Serial"
```

```

Sub Main
Dim objDrvInt As DriverInterface
Dim bRet As Boolean

    Set objDrvInt = GetDriverInterface(DRIVER_NAME)

    'SAVE ALL STATIONS
    bRet = objDrvInt.SaveConfigDriver

    If Not bRet Then Debug.Print "VBA Driver - SaveConfigDriver Error!"

    Set objDrvInt = Nothing

End Sub

```

StartStation, DriverInterface Function

Syntax StartStation(lpszName)

Description Loads and starts a thread of a new station dynamically added by the function AddStation.

Remarks To initiate communication of a station added by AddStation is necessary call the function startStation.

Please note. The stations created from the file .DrvSettings not need to be initiated by startStation.

| Parameter | Description |
|--------------------|---------------|
| lpszName As String | Station Name. |

Result Boolean
If the function is successful it returns the value True.

Example:

```

Option Explicit
Const DRIVER_NAME = "Modbus Serial"

Sub Main
Dim objDrvInt As DriverInterface
Dim sStationNameas String
Dim bRet As Boolean

    sStationName = This.GetParameter(0)

    If sStationName <> "" Then

        Set objDrvInt = GetDriverInterface(DRIVER_NAME)

        'START STATION
        bRet = objDrvInt.StartStation(sStationName)
    End If

    If Not bRet Then Debug.Print "VBA Driver - StartStation Error! Station
name: " & sStationName & ""

    Set objDrvInt = Nothing

End Sub

```

DriverInterface Properties

DelayEvents, DriverInterface Property

| | |
|--------------------|---|
| Syntax | DelayEvents = Long |
| Description | Gets or sets the delay for DriverInterface events. |
| Remarks | Only available from external COM clients. Not available from Supervisor BasicScripts. |

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long
Events delay interval in milliseconds.

Example:

```
Private Sub Form_Load()  
    Dim INewVal As Long  
    Dim bRet As Boolean  
    Dim drv As Object  
  
    Set drv = GetDriverInterface("ModbusTCP/IP")  
    INewVal = drv.DelayEvents  
    INewVal = INewVal + 100  
    drv.DelayEvents = INewVal  
End Sub
```

1.6.3. TaskInterface

TaskInterface Functions

Execute, TaskInterface Function

| | |
|--------------------|--|
| Syntax | Execute(dwTimeout) |
| Description | <p>This function puts Tasks into execution. The task in question can previously be created with the "AddDriverTask" function, or can be an already existing task from the driver's static task list and must be referenced by calling the "GetDriverTask" function.</p> <p>ATTENTION! This function returns the True value in cases where the task has been executed, however in order to know if it has been executed successfully without errors, you will need to test the station's status.</p> |
| Remarks | |

| Parameter | Description |
|-------------------|---|
| DwTimeout As Long | Time (milliseconds) to wait for task completion. Setting this parameter with a value higher than zero (it is advised that you set a value higher than the one set in the station's timeout time) the task will be executed in Synchronous mode. Setting this parameter to Zero, the task will be executed in Asynchronous mode. In both cases, however, the task will be executed only once, therefore if you use Asynchron Tasks the code must be created so that the function call is recycled. |

Result Boolean
True = the function has been executed successfully

False = function execution failed

Example:

See the "AddDriverTask" and "**GetDriverTask**" function example.

GetFlatVariableList, TaskInterface Function

Syntax GetFlatVariableList

Description Gets the Variable List of a static task. As only static tasks have a Variable List, this function is useless with dynamic tasks.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String
A String containing the Variable List. The separator character can be retrieved by calling the "GetVariableListSeparator" function.

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim strVarlist As String

    Set drv = GetDriverInterface("Modbus TCPIP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    strVarlist = task.GetFlatVariableList
    MsgBox "GetFlatVariableList = " & strVarlist, vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub
```

GetMaxByteSize, TaskInterface Function

Syntax GetMaxByteSize

Description Gets the maximum number of bytes that can be allocated for a task's read/write buffers.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
```

```

Dim nSizeBytes As Long

Set drv = GetDriverInterface("Modbus TCP/IP")
Set task = drv.GetDriverTask("Task")
If task Is Nothing Then
    MsgBox "Task not found!", vbExclamation, "ERROR "
Exit Sub
End If

nSizeBytes = task.GetMaxByteSize
MsgBox "GetMaxByteSize = " & CStr(nSizeBytes), vbInformation, "ERROR "

Set drv = Nothing
Set task = Nothing
End Sub

```

GetReadByteBuffer, TaskInterface Function

Syntax GetReadByteBuffer

Description Returns the latest read values from the task. The Task must previously be added by calling the "AddDriverTask" function and at least one call to the "Execute" function must have been made. If task is already present in the project's static task list, it will not be necessary to create and execute the task but it will have to be referenced only by calling the "GetDriverTask" function.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Variant
A Variant containing the last read values from the task. Usually the variable data type is dimensioned by the driver based on the data area to be read. If The task is a static task then the function will retrieve an Byte Array containing the data.

Example1:
See the "GetDriverTask" function example.

Example2:

```

Sub Main
    Dim drv As DriverInterface
    Dim Station As StationInterface
    Dim task As TaskInterface
    Dim ret As Boolean
    Dim var As Variant

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.AddDriverTask("Async", "Sta=Station1|Unit=1|FC=2|SA=2")
    If Not task.IsValid Then
        MsgBox "Task Async is not valid!", vbExclamation, "ERROR"
Exit Sub
    End If

    task.PollingTime = 500

    While Not This.IsStopping
        ret = task.Execute(5000)
        If ret = False Then
            MsgBox "task.Execute failed!", vbExclamation, "ERROR"
Exit While
        End If

        var = task.GetReadByteBuffer
        SetVariableValue("AsyncRead", var)
        DoEvents
    Wend

```

```

        Set Station = Nothing
        Set drv = Nothing
        Set task = Nothing
    End Sub

```

GetStationObject, TaskInterface Function

Syntax GetStationObject

Description Gets the interface to a tasks's station object

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Object
 If Function has been executed successfully it will retrieve an object of type StationInterface if otherwise Nothing is returned.

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim Station As StationInterface
    Dim task As TaskInterface

    'Create driver object
    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Async")
    If task Is Nothing Then
        MsgBox "Async task not found!",vbExclamation,"ERROR "
        Exit Sub
    End If

    'Create station object
    Set Station = task.GetStationObject
    If Station.IsInError Then
        MsgBox "Station in Error!",vbExclamation,"ERROR "
        Exit Sub
    End If

    Set Station = Nothing
    Set drv = Nothing
    Set task = Nothing
End Sub

```

GetTotalSizeBytes, TaskInterface Function

Syntax GetTotalSizeBytes

Description Gets the number of bytes allocated for a task's read/write buffers.

Remarks If bit variables (being the Project RealTimeDB variables) have been used in the task, the returned value is a bit number (the driver reserves a byte for each bit internally).

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim nSizeBytes As Long

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    nSizeBytes = task.GetTotalSizeBytes
    MsgBox "GetTotalSizeBytes = " & CStr(nSizeBytes), vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub

```

GetVariableListSeparator, TaskInterface Function

Syntax GetVariableListSeparator

Description Gets the Variable List Separator of a static task. As only static tasks have a Variable List, this function is useless with dynamic tasks.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String
A String containing the Variable List separator character (default is ';').

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim strListSep As String

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    strListSep = task.GetVariableListSeparator
    MsgBox "GetVariableListSeparator = " & strListSep, vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub

```

GetXMLSettings, TaskInterface Function

Syntax GetXMLSettings

Description Gets all settings of a task in XML format.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim strGetXMLSettings As String

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    strGetXMLSettings = task.GetXMLSettings
    MsgBox "GetXMLSettings = " & strGetXMLSettings, vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub
```

HasBeenExecuted, TaskInterface Function

Syntax HasBeenExecuted

Description This function returns information that task has been executed in read, with success, at least once. This function is therefore used for "Input" or "Input/Output" tasks.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
 True = task has been executed in read at least once successfully
 False = task has still not been executed in read or if executed has returned an error

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim bHasBeenExecuted As Boolean

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    bHasBeenExecuted = task.HasBeenExecuted
    MsgBox "HasBeenExecuted = " & CStr(bHasBeenExecuted), vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub
```

IsValid, TaskInterface Function

Syntax IsValid

Description Returns information about whether a task is valid.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
True = task is valid
False = task is not valid

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim bIsValid As Boolean

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
    End If

    bIsValid = task.IsValid
    MsgBox "IsValid = " & CStr(bIsValid), vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub
```

SetWriteByteBuffer, TaskInterface Function

Syntax SetWriteByteBuffer(vBuffer)

Description Sets the task with values to write. The Task must be previously be added by a call to **"AddDriverTask"** function and at least one call to the "Execute" function must have been made. If task is already present in the static task list of the project, is not necessary create an execute the task but it must only be referenced by a call to "GetDriverTask" function.

Remarks

| Parameter | Description |
|--------------------|--------------------------|
| VBuffer As Variant | The buffer to be written |

Result Boolean
True = the function has been executed successfully
False = function execution failed

Example:

See the "AddDriverTask" function example.

ToString, TaskInterface Function

Syntax ToString

Description Gets the Dynamic Settings string of a task. Please refer to each driver's specific documentation for Dynamic Settings formats.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String
String containing the Dynamic Settings for the task. Please refer to each driver's specific documentation for Dynamic Settings formats.

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim strToString As String

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    strToString = task.ToString
    MsgBox "ToString = " & strToString, vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub
```

TaskInterface Properties

AutoDelete, TaskInterface Property

Syntax Boolean = task.AutoDelete

Description Gets a task's AutoDelete flag.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
True = AutoDelete enabled
False = AutoDelete disabled

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim bAutoDelete As Boolean

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If
```

```

End If

bAutoDelete = task.AutoDelete
MsgBox "AutoDelete = " & CStr(bAutoDelete),vbInformation,"ERROR "

Set drv = Nothing
Set task = Nothing
End Sub

```

ConditionalVariable, TaskInterface Property

Syntax String = task.ConditionalVariable

Description Gets a task's conditional variable name.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim strCondVar As String

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!",vbExclamation,"ERROR"
        Exit Sub
    End If

    strCondVar = task.ConditionalVariable
    MsgBox "ConditionalVariable = " & strCondVar,vbInformation,"ERROR"

    Set drv = Nothing
    Set task = Nothing
End Sub

```

InUse, TaskInterface Property

Syntax Boolean = task.InUse

Description Gets a task's InUse flag.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
 True = task in use
 False = task not in use

Example:

```

Sub Main

```

```

Dim drv As DriverInterface
Dim task As TaskInterface
Dim bInUse As Boolean

Set drv = GetDriverInterface("Modbus TCP/IP")
Set task = drv.GetDriverTask("Task")
If task Is Nothing Then
    MsgBox "Task not found!", vbExclamation, "ERROR "
    Exit Sub
End If

bInUse = task.InUse
MsgBox "InUse = " & CStr(bInUse), vbInformation, "ERROR "

Set drv = Nothing
Set task = Nothing
End Sub

```

LastExecutionTime, TaskInterface Property

Syntax Date = task.LastExecutionTime

Description Gets a task's last execution time.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Date

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim dLastExec As Date

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    dLastExec = task.LastExecutionTime
    MsgBox "LastExecutionTime = " & CStr(dLastExec), vbInformation, "ERROR"

    Set drv = Nothing
    Set task = Nothing
End Sub

```

Name, TaskInterface Property

Syntax String = task.Name

Description Gets the name of a task.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

| | |
|------|------|
| None | None |
|------|------|

Result String

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim strName As String

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    strName = task.Name
    MsgBox "Task Name = " & strName, vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub
```

PollingTime, TaskInterface Property

Syntax Long = task.PollingTime

Description Gets a task's in use Polling Time. The time is in Milliseconds.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim lPollTime As Long

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    lPollTime = task.PollingTime
    MsgBox "PollingTime = " & CStr(lPollTime), vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub
```

PollingTimeNoInUse, TaskInterface Property

Syntax Long = task.PollingTimeNoInUse

Description Gets a task's "not-in-use" Polling Time. The time is in Milliseconds.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim IPollTime As Long

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    IPollTime = task.PollingTimeNoInUse
    MsgBox "PollingTimeNoInUse = " & CStr(IPollTime), vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub
```

Station, TaskInterface Property

Syntax String = task.Station

Description Gets the name of the task's station.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim strStation As String

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    strStation = task.Station
    MsgBox "Task Station = " & strStation, vbInformation, "ERROR "
```

```

        Set drv = Nothing
        Set task = Nothing
    End Sub

```

SwapBytes, TaskInterface Property

Syntax Boolean = task.SwapBytes

Description Gets a task's SwapBytes flag.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
 True = Swap Byte enabled
 False = Swap Byte disabled

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim bSwapBytes As Boolean

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    bSwapBytes = task.SwapBytes
    MsgBox "SwapBytes = " & CStr(bSwapBytes), vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub

```

SwapWords, TaskInterface Property

Syntax Boolean = task.SwapWords

Description Gets a task's SwapWords flag.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
 True = Swap Word enabled
 False = Swap Word disabled

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim bSwapWords As Boolean

```

```

Set drv = GetDriverInterface("Modbus TCPIP")
Set task = drv.GetDriverTask("Task")
If task Is Nothing Then
    MsgBox "Task not found!",vbExclamation,"ERROR "
Exit Sub
End If

bSwapBytes = task.SwapWords
MsgBox "SwapWords = " & CStr(bSwapWords),vbInformation,"ERROR "

Set drv = Nothing
Set task = Nothing
End Sub

```

Type, TaskInterface Property

Syntax Byte = task.Type

Description Gets a task's type.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Byte
0 = Input Type
1 = Input/Output Type
2 = Exception Output Type
3 = Unconditional Output Type
-2147220992 = not defined

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim nTaskType As Byte

    Set drv = GetDriverInterface("Modbus TCPIP")
    Set task = drv.GetDriverTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!",vbExclamation,"ERROR "
Exit Sub
End If

nTaskType = task.Type
MsgBox "Task Type = " & CStr(nTaskType),vbInformation,"ERROR "

Set drv = Nothing
Set task = Nothing
End Sub

```

WriteOutputsAtStartup, TaskInterface Property

Syntax Boolean = task.WriteOutputsAtStartup

Description This property returns the write settings of the task at project startup. Shows the value of the "Write Outputs at Startup" property. This only has meaning for "Input/Output" or "Exception Output" tasks.

Remarks Read only property.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

| | |
|------|------|
| None | None |
|------|------|

Result Boolean

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set task = drv.GetDriverTask("Task")
    If Not task.IsValid Then
        MsgBox "Task is not valid!", vbExclamation, "ERROR "
        Exit Sub
    End If
    MsgBox "WriteOutputsAtStartup = " & task.WriteOutputsAtStartup, vbExclamation, "ERROR "
    Set drv = Nothing
    Set task = Nothing
End Sub
```

1.6.4. StationInterface

StationInterface Functions

GetBaseStationInterface, StationInterface Function

Syntax GetBaseStationInterface

Description This function used in this interface return an object Nothing.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Nothing

Example:

GetSubStationInterface, StationInterface Function

Syntax GetSubStationInterface

Description Returns a SubStation object. According to the driver type, the returned object may be:
 "Nothing",
 "SerialStationInterface", "SerialHalfDuplexBaseStationInterface",
 "SerialTAPIHalfDuplexBaseStationInterface", "TAPIStationInterface",
 "TCPIPStationInterface", "RASStationInterface".

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Object:
 Nothing
 SerialStationInterface
 SerialHalfDuplexBaseStationInterface
 SerialTAPIHalfDuplexBaseStationInterface
 TCPIPStationInterface
 TAPIStationInterface for serial drivers
 RASStationInterface for TCPIP drivers

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim drv1 As DriverInterface
    Dim station1 As StationInterface
    Dim MODBUSStation As ModbusStationInterface
    Dim SerialTAPIHalfDuplexBaseStation As SerialTAPIHalfDuplexBaseStationInterface
    Dim SerialStation As SerialStationInterface
    Dim TAPIStation As TAPIStationInterface
    Dim TCPIPStation As TCPIPStationInterface
    Dim RASStation As RASStationInterface

    Set drv = GetDriverInterface("Modbus Serial")
    Set drv1 = GetDriverInterface("Modbus TCPIP")

    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set MODBUSStation = station.GetSubStationInterface
    If MODBUSStation Is Nothing Then
        MsgBox "MODBUSStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set SerialTAPIHalfDuplexBaseStation = MODBUSStation.GetBaseStationInterface
    If SerialTAPIHalfDuplexBaseStation Is Nothing Then
        MsgBox "SerialTAPIHalfDuplexBaseStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set TAPIStation = SerialTAPIHalfDuplexBaseStation.GetBaseStationInterface
    If TAPIStation Is Nothing Then
        MsgBox "TAPIStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set SerialStation = TAPIStation.GetBaseStationInterface
    If SerialStation Is Nothing Then
        MsgBox "SerialStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set station1 = drv1.GetDriverStation("Station2")
    If station1 Is Nothing Then
        MsgBox "Station1 not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set RASStation = station1.GetSubStationInterface
    If RASStation Is Nothing Then
        MsgBox "RASStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set TCPIPStation = RASStation.GetBaseStationInterface
    If TCPIPStation Is Nothing Then
        MsgBox "TCPIPStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If
```

```

Set drv = Nothing
Set station = Nothing
Set drv1 = Nothing
Set station1 = Nothing
Set TAPIStation = Nothing
Set SerialStation = Nothing
Set RASStation = Nothing
Set TCPIPStation = Nothing
End Sub

```

GetTask, StationInterface Function

Syntax GetTask(lpszTaskName)

Description Returns a task object. This function can be used to retrieve the interface to any static or dynamic task.

Remarks

| Parameter | Description |
|------------------------|----------------|
| lpszTaskName As String | Nome del Task. |

Result Object
If Function has been executed successfully it will retrieve an object of type TaskInterface if otherwise Nothing is returned.

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim task As TaskInterface
    Dim bInUse As Boolean

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    Set task = station.GetTask("Task")
    If task Is Nothing Then
        MsgBox "Task not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    bInUse = task.InUse
    MsgBox "InUse = " & CStr(bInUse), vbInformation, "ERROR "

    Set drv = Nothing
    Set station = Nothing
End Sub

```

GetXMLSettings, StationInterface Function

Syntax GetXMLSettings

Description Gets all settings of a station in XML format.

Remarks

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

| | |
|------|------|
| None | None |
|------|------|

Result String

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim strGetXMLSettings As String

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    strGetXMLSettings = station.GetXMLSettings
    MsgBox "GetXMLSettings = " & strGetXMLSettings, vbInformation, "ERROR "

    Set drv = Nothing
    Set station = Nothing
End Sub
```

IsInError, StationInterface Function

Syntax IsInError

Description Returns information about whether a station is in error.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
 True = station in error
 False = Station ok

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim bError As Boolean

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    bError = station.IsInError
    MsgBox "IsInError = " & CStr(bError), vbInformation, "ERROR "

    Set drv = Nothing
    Set station = Nothing
End Sub
```

StationInterface Properties

LastErrorCode, StationInterface Property

Syntax Long = task.LastErrorCode

Description Gets a station's last error code.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim lLastError As Long

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    lLastError = station.LastErrorCode
    MsgBox "LastErrorCode = " & CStr(lLastError), vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub
```

LastErrorString, StationInterface Property

Syntax String = task.LastErrorString

Description Gets a station's last error description string.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim lLastError As String

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
```

```

        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    ILastError = station.LastErrorString
    MsgBox "LastErrorString = " & CStr(ILastError), vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub

```

Name, StationInterface Property

Syntax String = task.Name

Description Gets a station's name.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim strName As String

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    strName = station.Name
    MsgBox "Name = " & strName, vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub

```

ErrorThreshold, StationInterface Property

Syntax Long = station.ErrorThreshold

Description This property sets or returns the parameter of the station: Error Threshold.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim lErrorThreshold As Long

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    station.ErrorThreshold = 2

    lErrorThreshold = station.ErrorThreshold
    MsgBox "Error Threshold = " & lErrorThreshold, vbInformation, "ERROR"

    Set drv = Nothing
    Set task = Nothing
End Sub

```

StateCommandVariable, StationInterface Property

Syntax String = station.StateCommandVariable

Description This property sets or returns the parameter of the station: State/Command Variable.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim strStateCommandVariable As String

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    station.StateCommandVariable = "StateCommand"

    strStateCommandVariable = station.StateCommandVariable
    MsgBox "State/Command Variable = " & strStateCommandVariable, vbInformation, "ERROR"

    Set drv = Nothing
    Set task = Nothing
End Sub

```

Quality, StationInterface Property

Syntax Integer = task.Quality

Description Gets a station's quality state.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Integer
The returned values tally with the OPC specification quality values:

252 = OPC_STATUS_MASK
3 = OPC_LIMIT_MASK
0 = OPC_QUALITY_BAD
64 = OPC_QUALITY_UNCERTAIN
192 = OPC_QUALITY_GOOD
4 = OPC_QUALITY_CONFIG_ERROR
8 = OPC_QUALITY_NOT_CONNECTED
12 = OPC_QUALITY_DEVICE_FAILURE
16 = OPC_QUALITY_SENSOR_FAILURE
20 = OPC_QUALITY_LAST_KNOWN
24 = OPC_QUALITY_COMM_FAILURE
28 = OPC_QUALITY_OUT_OF_SERVICE
68 = OPC_QUALITY_LAST_USABLE
80 = OPC_QUALITY_SENSOR_CAL
84 = OPC_QUALITY_EGU_EXCEEDED
88 = OPC_QUALITY_SUB_NORMAL
216 = OPC_QUALITY_LOCAL_OVERRIDE

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim nQuality As Integer

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    nQuality = station.Quality
    MsgBox "Quality = " & CStr(nQuality), vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub
```

StationType, StationInterface Property

Syntax Byte = task.StationType

Description Gets a station's type.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Byte
0 = normal. This value is returned when the driver uses external library, such as Modbus TCP/IP, INTERBUS, etc.
1 = SerialType

```

2 = TAPIType
3 = TCPIPType
4 = RASTCPIPType
-2147220992 = station not defined

```

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim task As TaskInterface
    Dim nStationType As Byte

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    nStationType = station.StationType
    MsgBox "StationType = " & CStr(nStationType), vbInformation, "ERROR "

    Set drv = Nothing
    Set task = Nothing
End Sub

```

1.6.5. SerialTAPIHalfDuplexBaseStationInterface

Functions

GetBaseStationInterface, SerialTAPIHalfDuplexBaseStationInterface Function

Syntax GetBaseStationInterface

Description This function used in this interface return an object "**TAPIStationInterface**".

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Object
If Function has been executed successfully it will retrieve an object of type **TAPIStationInterface**if , otherwise Nothing is returned.

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim MODBUSStation As ModbusStationInterface
    Dim TAPIHalfDupStation As SerialTAPIHalfDuplexBaseStationInterface
    Dim SerialStation As SerialStationInterface
    Dim TAPIStation As TAPIStationInterface
    Dim SubTAPIStation As TAPIStationInterface

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set MODBUSStation = station.GetSubStationInterface
    If MODBUSStation Is Nothing Then

```

```

        MsgBox "MODBUSStation not found!",vbExclamation,"ERROR"
    Exit Sub
End If

Set TAPIHalfDupStation = MODBUSStation .GetBaseStationInterface
If TAPIHalfDupStation Is Nothing Then
    MsgBox "TAPIHalfDupStation not found!",vbExclamation,"ERROR"
    Exit Sub
End If

Set TAPIStation= TAPIHalfDupStation .GetBaseStationInterface
If TAPIStationIs Nothing Then
    MsgBox "TAPIStationnot found!",vbExclamation,"ERROR"
    Exit Sub
End If

Set SerialStation = TAPIStation.GetBaseStationInterface
If SerialStation Is Nothing Then
    MsgBox "SerialStation not found!",vbExclamation,"ERROR"
    Exit Sub
End If

Set SubTAPIStation = TAPIStation.GetSubStationInterface
If SubTAPIStation Is Nothing Then
    MsgBox "SubTAPIStation not found!",vbExclamation,"ERROR"
    Exit Sub
End If

Set drv = Nothing
Set station = Nothing
Set MODBUSStation = Nothing
Set TAPIStation = Nothing
Set SerialStation = Nothing
Set SubSerialStation = Nothing
Set SubTAPIStation = Nothing
End Sub

```

GetSubStationInterface, SerialTAPIHalfDuplexBaseStationInterface Function

Syntax GetSubStationInterface

Description This function used in this interface return an object **"SerialTAPIHalfDuplexBaseStationInterface"**.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Object
If Function has been executed successfully it will retrieve an object of type **SerialTAPIHalfDuplexBaseStationInterface** if , otherwise Nothing is returned.

Example:

Sub Main

```

Dim drv As DriverInterface
Dim station As StationInterface
Dim MODBUSStation As ModbusStationInterface
Dim SerialTAPIHalfDuplexBaseStation As SerialTAPIHalfDuplexBaseStationInterface

```

```

Set drv = GetDriverInterface("Modbus Serial")
Set station = drv.GetDriverStation("Station1")
If station Is Nothing Then
    MsgBox "Station not found!",vbExclamation,"ERROR"

```

```

Exit Sub
End If

Set MODBUSStation = station.GetSubStationInterface
If MODBUSStation Is Nothing Then
    MsgBox "MODBUSStation not found!",vbExclamation,"ERROR"
Exit Sub
End If

Set SerialTAPIHalfDuplexBaseStation = MODBUSStation .GetBaseStationInterface
If SerialTAPIHalfDuplexBaseStation Is Nothing Then
    MsgBox "SerialTAPIHalfDuplexBaseStation not found!",vbExclamation,"ERROR"
Exit Sub
End If

Set SerialTAPIHalfDuplexBaseStation = SerialTAPIHalfDuplexBaseStation .GetSubStationInterface
If SerialTAPIHalfDuplexBaseStation Is Nothing Then
    MsgBox "SerialTAPIHalfDuplexBaseStation not found!",vbExclamation,"ERROR"
Exit Sub
End If

Set drv = Nothing
Set station = Nothing
Set MODBUSStation = Nothing

End Sub

```

GetXMLSettings, SerialTAPIHalfDuplexBaseStationInterfaceFunction

Syntax GetXMLSettings

Description Gets all settings of a station in XML format.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim MODBUSStation As ModbusStationInterface
    Dim TAPIHalfDupStation As SerialTAPIHalfDuplexBaseStationInterface
    Dim strGetXMLSettings As String

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!",vbExclamation,"ERROR"
Exit Sub
    End If

    Set MODBUSStation = station.GetSubStationInterface
    If MODBUSStation Is Nothing Then
        MsgBox "MODBUSStation not found!",vbExclamation,"ERROR"
Exit Sub
    End If

    Set TAPIHalfDupStation = MODBUSStation .GetBaseStationInterface
    If TAPIHalfDupStation Is Nothing Then

```

```

        MsgBox "TAPIHalfDupStation not found!",vbExclamation,"ERROR"
    Exit Sub
End If

strGetXMLSettings = TAPIHalfDupStation .GetXMLSettings
MsgBox "GetXMLSettings = " & CStr(strGetXMLSettings),vbInformation,"ERROR"

Set drv = Nothing
Set station = Nothing
Set TAPIStation = Nothing
End Sub

```

Properties

DelayAfterLastChar, SerialTAPIHalfDuplexBaseStationInterface Property

| | |
|--------------------|---|
| Syntax | objTAPIHalfDpStation. DelayAfterLastChar |
| Description | This property sets or returns the time between the transmission last character and the setting to OFF control signal. The time is in milliseconds. |
| Remarks | This time should be considered non-deterministic, because Windows is not an OS real-time. |

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the example of the property "DelayBeforeFirstChar ".

DelayBeforeFirstChar, SerialTAPIHalfDuplexBaseStationInterface Property

| | |
|--------------------|--|
| Syntax | objTAPIHalfDpStation. DelayBeforeFirstChar |
| Description | This property sets or returns the time between the set to the ON control signal and the transmission of a message. The time is expressed in milliseconds. |
| Remarks | This time should be considered non-deterministic, because Windows is not an OS real-time. |

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```

Option Explicit
Const DRIVER_NAME = "Modbus Serial"

Dim objDrvInt As DriverInterface
Dim objStation As StationInterface
Dim objModbusStation As ModbusStationInterface
Dim objTAPIHalfDpStation As SerialTAPIHalfDuplexBaseStationInterface

```

```

Enum eHafDupSignal
    None
    RTS
    DTS
End Enum

Sub Main
Dim objDrvInt As DriverInterface
Dim sStationName as String
    sStationName = This.GetParameter(0)
    If sStationName <> "" Then
        Set objDrvInt = GetDriverInterface(DRIVER_NAME)
        Set objStation = objDrvInt.GetDriverStation(sStationName)
        If Not objStation Is Nothing Then
            Set objModbusStation = objStation.GetSubStationInterface
            'Parametrize Modbus Station properties
            If Not objModbusStation Is Nothing Then
                'Retreive the TAPI Half-Duplex Base Objects
                Set objTAPIHalfDpStation = objModbusStation.GetBaseStationInterface
                If Not objTAPIHalfDpStation Is Nothing Then
                    objTAPIHalfDpStation.DelayBeforeFirstChar = 0
                Else
                    Debug.Print "VBA Driver - objTAPIHalfDpStation Error! Station: " &
objStation.Name & "
                End If
            Else
                Debug.Print "VBA Driver - objModbusStation is Nothing for the Station" &
sStationName & ""
            End If
        Else
            Debug.Print "VBA Driver - objStation is Nothing for the Station" & sStationName & ""
        End If
        Set objDrvInt = Nothing
        Set objStation= Nothing
        Set objModbusStation = Nothing
        Set objTAPIHalfDpStation = Nothing
    End Sub

```

HalfDuplexSignal, SerialTAPIHalfDuplexBaseStationInterface Property

Syntax objTAPIHalfDpStation.**HalfDuplexSignal**

Description This property selects or returns the control signal used to control the direction of transmission.
Possible values: None, RTS and DTR.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the example of the property "DelayBeforeFirstChar ".

1.6.6. SerialStationInterface

SerialStationInterface Functions

GetSubStationInterface, SerialStationInterface Function

Syntax GetSubStationInterface

Description This function, according to driver type, used at this level may return **"SerialStationInterface"** or **"ModbusStationInterface"** object type.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Object:
SerialStationInterface
ModbusStationInterface

Example:
See the "GetBaseStationInterface" property example.

GetXMLSettings, SerialStationInterface Function

Syntax GetXMLSettings

Description Gets all settings of a station in XML format.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:
Sub Main
 Dim drv As DriverInterface
 Dim station As StationInterface
 Dim SerialStation As SerialStationInterface
 Dim TAPIStation As TAPIStationInterface
 Dim strGetXMLSettings As String

 Set drv = GetDriverInterface("Modbus Serial")
 Set station = drv.GetDriverStation("Station1")
 If station Is Nothing Then
 MsgBox "Station not found!", vbExclamation, "ERROR "
 Exit Sub
 End If

 Set TAPIStation = station.GetSubStationInterface
 If TAPIStation Is Nothing Then
 MsgBox "TAPIStation not found!", vbExclamation, "ERROR "
 Exit Sub
 End If

 Set SerialStation = TAPIStation.GetBaseStationInterface

```

If SerialStation Is Nothing Then
    MsgBox "SerialStation not found!",vbExclamation,"ERROR "
    Exit Sub
End If

strGetXMLSettings = SerialStation.GetXMLSettings
MsgBox "GetXMLSettings = " & CStr(strGetXMLSettings),vbInformation,"ERROR "

Set drv = Nothing
Set station = Nothing
Set TAPIStation = Nothing
Set SerialStation = Nothing
End Sub

```

SerialStationInterface Properties

BaudRate, SerialStationInterface Property

Syntax Long = station.BaudRate

Description This property sets or returns the COM Port BaudeRate value of the Station.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim MODBUSStation As ModbusStationInterface
    Dim TAPIHalfDupStation As SerialTAPIHalfDuplexBaseStationInterface
    Dim SerialStation As SerialStationInterface
    Dim TAPIStation As TAPIStationInterface
    Dim lBaudRate As Long
    Dim nParity As Byte
    Dim nPortID As Byte
    Dim nStopBits As Byte
    Dim nByteSize As Byte
    Dim nFlowControl As Byte

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!",vbExclamation,"ERROR "
        Exit Sub
    End If

    Set MODBUSStation = station.GetSubStationInterface
    If MODBUSStation Is Nothing Then
        MsgBox "MODBUSStation not found!",vbExclamation,"ERROR"
        Exit Sub
    End If

    Set SerialTAPIHalfDuplexBaseStation = MODBUSStation .GetBaseStationInterface
    If SerialTAPIHalfDuplexBaseStation Is Nothing Then
        MsgBox "SerialTAPIHalfDuplexBaseStation not found!",vbExclamation,"ERROR"
        Exit Sub
    End If

```

```

Set TAPIStation = SerialTAPIHalfDuplexBaseStation .GetBaseStationInterface
If TAPIStation Is Nothing Then
    MsgBox "TAPIStation not found!",vbExclamation,"ERROR"
Exit Sub
End If

Set SerialStation = TAPIStation.GetBaseStationInterface
If SerialStation Is Nothing Then
    MsgBox "SerialStation not found!",vbExclamation,"ERROR "
Exit Sub
End If

nPortID = SerialStation.PortID
lBaudRate = SerialStation.BaudRate
nByteSize = SerialStation.ByteSize
nParity = SerialStation.Parity
nStopBits = SerialStation.StopBits
nFlowControl = SerialStation.FlowControl

MsgBox "PortID = " & CStr(nPortID) & vbCrLf & _
"BaudRate = " & CStr(lBaudRate) & vbCrLf & _
"ByteSize = " & CStr(nByteSize) & vbCrLf & _
"Parity = " & CStr(nParity) & vbCrLf & _
"StopBits = " & CStr(nStopBits) & vbCrLf & _
"FlowControl = " & CStr(nFlowControl),vbInformation,"ERROR "

Set drv = Nothing
Set station = Nothing
Set TAPIStation = Nothing
Set SerialStation = Nothing
End Sub

```

ByteSize, SerialStationInterface Property

Syntax Byte = station.ByteSize

Description This property sets or returns the COM Port ByteSize value of the Station.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Byte

Example:

See the "BaudRate" property example.

CdTimeout, SerialStationInterface Property

Syntax Long = station.CdTimeout

Description This property sets or returns the Timeout for the COM Port Cd signal of the Station. This time is in milliseconds.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the "RxTimeout" property example.

CtsTimeout, SerialStationInterface Property

Syntax Long = station.CtsTimeout

Description This property sets or returns the Timeout for the COM Port Cts signal of the Station. This time is in milliseconds.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the "RxTimeout" property example.

DsrTimeout, SerialStationInterface Property

Syntax Long = station.DsrTimeout

Description This property sets or returns the Timeout for the COM Port Dsr signal of the Station. This time is in milliseconds.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the "RxTimeout" property example.

FlowControl, SerialStationInterface Property

Syntax Byte = station.FlowControl

Description This property sets or returns the COM Port Flow Control value of the Station.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Byte
0 = None

1 = Hardware
2 = Xon / Xoff

Example:

See the "BaudRate" property example.

KeepPortOpened, SerialStationInterface Property

Syntax Boolean = station.KeepPortOpened

Description This property allow to keep the COM Port always opened during the project runtime. If the property is set to false the COM port will be opened and then will be closed each time that some data have to be exchanged.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
True = COM port is keep opened
False = COM port is opened and closed each time some data have to be exchanged

Example:

Sub Main

```
Dim drv As DriverInterface
Dim station As StationInterface
Dim SerialStation As SerialStationInterface
Dim MODBUSStation As ModbusStationInterface
Dim TAPIHalfDupStation As SerialTAPIHalfDuplexBaseStationInterface
Dim TAPIStation As TAPIStationInterface
Dim bKeepPortOpened As Boolean

Set drv = GetDriverInterface("Modbus Serial")
Set station = drv.GetDriverStation("Station1")
If station Is Nothing Then
    MsgBox "Station not found!", vbExclamation, "ERROR"
    Exit Sub
End If

Set MODBUSStation = station.GetSubStationInterface
If MODBUSStation Is Nothing Then
    MsgBox "MODBUSStation not found!", vbExclamation, "ERROR"
    Exit Sub
End If

Set SerialTAPIHalfDuplexBaseStation = MODBUSStation .GetBaseStationInterface
If SerialTAPIHalfDuplexBaseStation Is Nothing Then
    MsgBox "SerialTAPIHalfDuplexBaseStation not found!", vbExclamation, "ERROR"
    Exit Sub
End If

Set TAPIStation = SerialTAPIHalfDuplexBaseStation .GetBaseStationInterface
If TAPIStation Is Nothing Then
    MsgBox "TAPIStation not found!", vbExclamation, "ERROR"
    Exit Sub
End If

Set SerialStation = TAPIStation.GetBaseStationInterface
If SerialStation Is Nothing Then
    MsgBox "SerialStation not found!", vbExclamation, "ERROR"
    Exit Sub
End If

bKeepPortOpened = SerialStation.KeepPortOpened
MsgBox "KeepPortOpened = " & CStr(bKeepPortOpened), vbInformation, "ERROR"
```

```

Set drv = Nothing
Set station = Nothing
Set TAPIStation = Nothing
Set SerialStation = Nothing
End Sub

```

Parity, SerialStationInterface Property

Syntax Byte = station.Parity

Description This property sets or returns the COM Port Parity value of the Station.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Byte
0 = None
1 = Odd
2 = Even
3 = Mark
4 = Blank

Example:
See the "BaudRate" property example.

PortID, SerialStationInterface Property

Syntax Byte = station.PortID

Description This property sets or returns the COM Port number of the Station.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Byte
0 = COM1
1 = COM2
2 = COM3
3 = COM4
.....

Example:
See the "BaudRate" property example.

RxTimeout, SerialStationInterface Property

Syntax Long = station.RxTimeout

Description This property sets or returns the Timeout for the COM Port Rx signal of the Station. This time is in milliseconds.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim SerialStation As SerialStationInterface
    Dim MODBUSStation As ModbusStationInterface
    Dim TAPIHalfDupStation As SerialTAPIHalfDuplexBaseStationInterface
    Dim TAPIStation As TAPIStationInterface
    Dim IRxTimeout As Long
    Dim ITxTimeout As Long
    Dim ICdTimeout As Long
    Dim ICtsTimeout As Long
    Dim IDsrTimeout As Long

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set MODBUSStation = station.GetSubStationInterface
    If MODBUSStation Is Nothing Then
        MsgBox "MODBUSStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set SerialTAPIHalfDuplexBaseStation = MODBUSStation.GetBaseStationInterface
    If SerialTAPIHalfDuplexBaseStation Is Nothing Then
        MsgBox "SerialTAPIHalfDuplexBaseStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set TAPIStation = SerialTAPIHalfDuplexBaseStation.GetBaseStationInterface
    If TAPIStation Is Nothing Then
        MsgBox "TAPIStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set SerialStation = TAPIStation.GetBaseStationInterface
    If SerialStation Is Nothing Then
        MsgBox "SerialStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    IRxTimeout = SerialStation.RxTimeout
    ITxTimeout = SerialStation.TxTimeout
    ICdTimeout = SerialStation.CdTimeout
    ICtsTimeout = SerialStation.CtsTimeout
    IDsrTimeout = SerialStation.DsrTimeout

    MsgBox "RxTimeout = " & CStr(IRxTimeout) & vbCrLf & _
        "TxTimeout = " & CStr(ITxTimeout) & vbCrLf & _
        "CdTimeout = " & CStr(ICdTimeout) & vbCrLf & _
        "CtsTimeout = " & CStr(ICtsTimeout) & vbCrLf & _
        "DsrTimeout = " & CStr(IDsrTimeout), vbInformation, "ERROR"

    Set drv = Nothing
    Set station = Nothing
    Set TAPIStation = Nothing
    Set SerialStation = Nothing
End Sub

```

SizeReceivingQueue, SerialStationInterface Property

Syntax Long = station.SizeReceivingQueue

Description This property sets or returns the Receiving Buffer Queue (Bytes) for the COM Port of the Station.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the "BaudRate" property example.

SizeTransmissionQueue, SerialStationInterface Property

Syntax Long = station.SizeTransmissionQueue

Description This property sets or returns the Transmission Buffer Queue (Bytes) for the COM Port of the Station.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the "BaudRate" property example.

StopBits, SerialStationInterface Property

Syntax Byte = station.StopBits

Description This property sets or returns the COM Port Stop Bits value of the Station.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Byte
0 = 1 stop bit
1 = 1.5 stop bits
2 = 2 stop bits

Example:

See the "BaudRate" property example.

TxTimeout, SerialStationInterface Property

Syntax Long = station.TxTimeout

Description This property sets or returns the Timeout for the COM Port Tx signal of the Station. This time is in milliseconds.

Remarks A modification of this value will come acquired only to the following serial port opening.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the "RxTimeout" property example.

1.6.7. TAPIStationInterface

TAPIStationInterface Functions

GetBaseStationInterface, TAPIStationInterface Function

Syntax GetBaseStationInterface

Description This function used in this interface return an object "**SerialStationInterface**".

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Object
If Function has been executed successfully it will retrieve an object of type SerialStationInterface if , otherwise Nothing is returned.

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim MODBUSStation As ModbusStationInterface
    Dim TAPIHalfDupStation As SerialTAPIHalfDuplexBaseStationInterface
    Dim SerialStation As SerialStationInterface
    Dim TAPIStation As TAPIStationInterface
    Dim SubTAPIStation As TAPIStationInterface

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR"
    End If

    Set MODBUSStation = station.GetSubStationInterface
    If MODBUSStation Is Nothing Then
        MsgBox "MODBUSStation not found!", vbExclamation, "ERROR"
```

```

Exit Sub
End If

Set TAPIHalfDupStation = MODBUSStation .GetBaseStationInterface
If TAPIHalfDupStation Is Nothing Then
    MsgBox "TAPIHalfDupStation not found!",vbExclamation,"ERROR"
Exit Sub
End If

Set TAPIStation= TAPIHalfDupStation .GetBaseStationInterface
If TAPIStationIs Nothing Then
    MsgBox "TAPIStationnot found!",vbExclamation,"ERROR"
Exit Sub
End If

Set SerialStation = TAPIStation.GetBaseStationInterface
If SerialStation Is Nothing Then
    MsgBox "SerialStation not found!",vbExclamation,"ERROR"
Exit Sub
End If

Set SubTAPIStation = TAPIStation.GetSubStationInterface
If SubTAPIStation Is Nothing Then
    MsgBox "SubTAPIStation not found!",vbExclamation,"ERROR"
Exit Sub
End If

Set drv = Nothing
Set station = Nothing
Set TAPIStation = Nothing
Set SerialStation = Nothing
Set SubTAPIStation = Nothing
End Sub

```

GetSubStationInterface, TAPIStationInterface Function

Syntax GetSubStationInterface

Description This function used in this interface return an object "**TAPIStationInterface**".

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Object
If Function has been executed successfully it will retrieve an object of type TAPIStationInterface if , otherwise Nothing is returned.

Example:
See the "GetBaseStationInterface" property example.

GetXMLSettings, TAPIStationInterface Function

Syntax GetXMLSettings

Description Gets all settings of a station in XML format.

Remarks

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

| | |
|------|------|
| None | None |
|------|------|

Result String

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim MODBUSStation As ModbusStationInterface
    Dim TAPIHalfDupStation As SerialTAPIHalfDuplexBaseStationInterface
    Dim TAPIStation As TAPIStationInterface
    Dim strGetXMLSettings As String

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set MODBUSStation = station.GetSubStationInterface
    If MODBUSStation Is Nothing Then
        MsgBox "MODBUSStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set TAPIHalfDupStation = MODBUSStation .GetBaseStationInterface
    If TAPIHalfDupStation Is Nothing Then
        MsgBox "TAPIHalfDupStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set TAPIStation = TAPIHalfDupStation .GetBaseStationInterface
    If TAPIStation Is Nothing Then
        MsgBox "TAPIStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    strGetXMLSettings = TAPIStation.GetXMLSettings
    MsgBox "GetXMLSettings = " & CStr(strGetXMLSettings), vbInformation, "ERROR"

    Set drv = Nothing
    Set station = Nothing
    Set TAPIStation = Nothing
End Sub
```

IsConnected, TAPIStationInterface Function

Syntax IsConnected

Description Gets the connection state of the RAS Station. The "True" value indicates that the station is being connected or already connected.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
True = Station Connected or being connected
False = Station not Connected

Example:

```
Sub Main
```

```

Dim drv As DriverInterface
Dim station As StationInterface
Dim MODBUSStation As ModbusStationInterface
Dim TAPIHalfDupStation As SerialTAPIHalfDuplexBaseStationInterface
Dim TAPIStation As TAPIStationInterface
Dim bIsConnected As Boolean

Set drv = GetDriverInterface("Modbus Serial")
Set station = drv.GetDriverStation("Station1")
If station Is Nothing Then
    MsgBox "Station not found!", vbExclamation, "ERROR"
    Exit Sub
End If

Set MODBUSStation = station.GetSubStationInterface
If MODBUSStation Is Nothing Then
    MsgBox "MODBUSStation not found!", vbExclamation, "ERROR"
    Exit Sub
End If

Set TAPIHalfDupStation = MODBUSStation .GetBaseStationInterface
If TAPIHalfDupStation Is Nothing Then
    MsgBox "TAPIHalfDupStation not found!", vbExclamation, "ERROR"
    Exit Sub
End If

Set TAPIStation= TAPIHalfDupStation .GetBaseStationInterface
If TAPIStationIs Nothing Then
    MsgBox "TAPIStationnot found!", vbExclamation, "ERROR"
    Exit Sub
End If

bIsConnected = TAPIStation.IsConnected
MsgBox "IsConnected = " & CStr(bIsConnected), vbInformation, "ERROR"

Set drv = Nothing
Set station = Nothing
Set TAPIStation = Nothing
End Sub

```

TAPIStationInterface Properties

DisconnectAfterSecs, TAPIStationInterface Property

Syntax Long = station.DisconnectAfterSecs

Description This property sets or returns the time in milliseconds after which the modem connection of the TAPI Station will hang up. The counter of the time will start from the moment in which all the communication tasks are not in use, and it will be reset if at least one task returns in use before the expiration of the time.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim MODBUSStation As ModbusStationInterface
    Dim TAPIHalfDupStation As SerialTAPIHalfDuplexBaseStationInterface

```

```

Dim TAPIStation As TAPIStationInterface
Dim strPhoneNumber As String
Dim nRetries As Byte
Dim IDisconAfter As Long
Dim IRetryAfter As Long
Dim bEnableTAPI As Boolean
Dim bPrompt As Boolean
Dim bShowCon As Boolean

Set drv = GetDriverInterface("Modbus Serial")
Set station = drv.GetDriverStation("Station1")
If station Is Nothing Then
    MsgBox "Station not found!", vbExclamation, "ERROR "
    Exit Sub
End If

Set MODBUSStation = station.GetSubStationInterface
If MODBUSStation Is Nothing Then
    MsgBox "MODBUSStation not found!", vbExclamation, "ERROR"
    Exit Sub
End If

Set TAPIHalfDupStation = MODBUSStation .GetBaseStationInterface
If TAPIHalfDupStation Is Nothing Then
    MsgBox "TAPIHalfDupStation not found!", vbExclamation, "ERROR"
    Exit Sub
End If

Set TAPIStation= TAPIHalfDupStation .GetBaseStationInterface
If TAPIStationIs Nothing Then
    MsgBox "TAPIStationnot found!", vbExclamation, "ERROR"
    Exit Sub
End If

strPhoneNumber = TAPIStation.PhoneNumber
nRetries = TAPIStation.Retries
IDisconAfter = TAPIStation.DisconnectAfterSecs
IRetryAfter = TAPIStation.RetryAfterSecs
bEnableTAPI = TAPIStation.EnableTAPICallOnThisStation
bPrompt = TAPIStation.PromptForConnection
bShowCon = TAPIStation.ShowConnectionDlg

MsgBox "PhoneNumber = " & CStr(strPhoneNumber) & vbLf & _
    "Retries = " & CStr(nRetries) & vbLf & _
    "DisconnectAfterSecs = " & CStr(IDisconAfter) & vbLf & _
    "RetryAfterSecs = " & CStr(IRetryAfter) & vbLf & _
    "EnableTAPICallOnThisStation = " & CStr(bEnableTAPI) & vbLf & _
    "PromptForConnection = " & CStr(bPrompt) & vbLf & _
    "DsrTimShowConnectionDlgeout = " & CStr(bShowCon), vbInformation, "ERROR "

Set drv = Nothing
Set station = Nothing
Set TAPIStation = Nothing
End Sub

```

EnableTAPICallOnThisStation, TAPIStationInterface Property

- Syntax** Boolean = station.DisconnectAfterSecs
- Description** This property sets or returns the enable to execute the modem connection for the Station.
- Remarks** A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
True = TAPI call enable
False = TAPI call disable

Example:
See the "**DisconnectAfterSecs**" property example.

EndConnectionTime, TAPIStationInterface Property

Syntax Date = station.EndConnectionTime

Description This property returns the date and time of end of the last modem connection of the Station.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Date

Example:
See the "**StartConnectionTime**" property example.

LastConnectionTime, TAPIStationInterface Property

Syntax Date = station.LastConnectionTime

Description This property returns the connection time of the last modem call of the Station.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Date

Example:
See the "**StartConnectionTime**" property example.

LastTAPIError, TAPIStationInterface Property

Syntax Long = station.LastTAPIError

Description This property returns the last error code occurs during the modem call of the Station.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the "**StartConnectionTime**" property example.

LastTAPIErrorString, TAPIStationInterface Property

Syntax String = station.LastTAPIErrorString

Description This property returns a description text about the state of the modem call of the Station. This string can be an error message or a state connection message.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

See the "**StartConnectionTime**" property example.

PhoneNumber, TAPIStationInterface Property

Syntax String = station.PhoneNumber

Description This property sets or returns the telephone number to call.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

See the "**DisconnectAfterSecs**" property example.

PromptForConnection, TAPIStationInterface Property

Syntax Boolean = station.PromptForConnection

Description This property allows you to show or hide the connection confirmation dialog window. If this property is true, before start connection, the Supervisor will show a confirmation dialog window. Only if the user will click the OK button the Supervisor will start the connection.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
True = show the dialog window before connecting
False = don't show the dialog window and start connection

Example:
See the "**DisconnectAfterSecs**" property example.

Retries, TAPIStationInterface Property

Syntax Byte = station.Retries

Description This property sets or returns the number of retries to execute for the modem call of the Station when the connection fails.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Byte

Example:
See the "**DisconnectAfterSecs**" property example.

RetryAfterSecs, TAPIStationInterface Property

Syntax Long = station.RetryAfterSecs

Description This property sets or returns the time in milliseconds after which the modem connection will be retried if the previous call has failed.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:
See the "**DisconnectAfterSecs**" property example.

ShowConnectionDlg, TAPIStationInterface Property

Syntax Boolean = station.ShowConnectionDlg

Description This property allows you to show or hide the connection status dialog window.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
True = show dialog window
False = hide dialog window

Example:
See the "**DisconnectAfterSecs**" property example.

StartConnectionTime, TAPIStationInterface Property

Syntax Date = station.StartConnectionTime

Description This property returns the date and time of begin of the last modem connection of the Station.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Date

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim MODBUSStation As ModbusStationInterface
    Dim TAPIHalfDupStation As SerialTAPIHalfDuplexBaseStationInterface
    Dim TAPIStation As TAPIStationInterface
    Dim dStartConn As Date
    Dim dEndConn As Date
    Dim dTotalConn As Date
    Dim dLastConn As Date
    Dim lLastError As Long
    Dim strLastErrorString As String

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set MODBUSStation = station.GetSubStationInterface
    If MODBUSStation Is Nothing Then
        MsgBox "MODBUSStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set TAPIHalfDupStation = MODBUSStation .GetBaseStationInterface
    If TAPIHalfDupStation Is Nothing Then
        MsgBox "TAPIHalfDupStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    Set TAPIStation = TAPIHalfDupStation .GetBaseStationInterface
    If TAPIStation Is Nothing Then
        MsgBox "TAPIStation not found!", vbExclamation, "ERROR"
        Exit Sub
    End If

    dStartConn = TAPIStation.StartConnectionTime
    dEndConn = TAPIStation.EndConnectionTime
    dTotalConn = TAPIStation.TotalConnectionTime
    dLastConn = TAPIStation.LastConnectionTime
    lLastError = TAPIStation.LastTAPIError
    strLastErrorString = TAPIStation.LastTAPIErrorString
```

```

MsgBox "StartConnectionTime = " & CStr(dStartConn) & vbCrLf & _
      "EndConnectionTime = " & CStr(dEndConn) & vbCrLf & _
      "TotalConnectionTime = " & CStr(dTotalConn) & vbCrLf & _
      "LastConnectionTime = " & CStr(dLastConn) & vbCrLf & _
      "LastTAPIError = " & CStr(ILastError) & vbCrLf & _
      "LastTAPIErrorString = " & CStr(strLastErrorString),vbInformation,"ERROR"

Set drv = Nothing
Set station = Nothing
Set TAPIStation = Nothing
End Sub

```

TotalConnectionTime, TAPIStationInterface Property

Syntax Date = station.TotalConnectionTime

Description This property returns the total connection time of the modem call of the Station. The time is the sum of all calls executed from the start of the Project.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Date

Example:
See the "StartConnectionTime" property example.

1.6.8. TCPIPStationInterface

TCPIPStationInterface Functions

GetBaseStationInterface, TCPIPStationInterface Function

Syntax GetBaseStationInterface

Description This function used in this interface returns a "StationInterface" object.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Object
If Function has been executed successfully it will retrieve a StationInterface type object, otherwise Nothing is returned.

Example:
Sub Main
Dim drv As DriverInterface
Dim station As StationInterface
Dim RASStation As RASStationInterface
Dim SubRASStation As RASStationInterface
Dim TCPIPStation As TCPIPStationInterface
Dim SubTCPIPStation As StationInterface

```

Set drv = GetDriverInterface("Modbus TCP/IP")
Set station = drv.GetDriverStation("Station1")
If station Is Nothing Then
    MsgBox "Station not found!", vbExclamation, "ERROR "
Exit Sub
End If

Set RASStation = station.GetSubStationInterface
If RASStation Is Nothing Then
    MsgBox "RASStation not found!", vbExclamation, "ERROR "
Exit Sub
End If

Set TCPIPStation = RASStation.GetBaseStationInterface
If TCPIPStation Is Nothing Then
    MsgBox "TCPIPStation not found!", vbExclamation, "ERROR "
Exit Sub
End If

Set SubTCPIPStation = TCPIPStation.GetBaseStationInterface
If SubTCPIPStation Is Nothing Then
    MsgBox "SubTCPIPStation not found!", vbExclamation, "ERROR "
Exit Sub
End If

Set SubRASStation = TCPIPStation.GetSubStationInterface
If SubRASStation Is Nothing Then
    MsgBox "SubRASStation not found!", vbExclamation, "ERROR "
Exit Sub
End If

Set drv = Nothing
Set station = Nothing
Set RASStation = Nothing
Set TCPIPStation = Nothing
Set SubTCPIPStation = Nothing
Set SubRASStation = Nothing
End Sub

```

GetSubStationInterface, TCPIPStationInterface Function

Syntax GetSubStationInterface

Description This function, according to driver type, used at this level returns a **"TCPIPStationInterface"** or **"RASStationInterface"** object type.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Object:
TCPIPStationInterface
RASStationInterface

Example:
See the "GetBaseStationInterface" property example.

GetXMLSettings, TCPIPStationInterface Function

Syntax GetXMLSettings

Description Gets all settings of a station in XML format.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim RASStation As RASStationInterface
    Dim TCPIPStation As TCPIPStationInterface
    Dim strGetXMLSettings As String

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    Set RASStation = station.GetSubStationInterface
    If RASStation Is Nothing Then
        MsgBox "RASStation not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    Set TCPIPStation = RASStation.GetBaseStationInterface
    If TCPIPStation Is Nothing Then
        MsgBox "TCPIPStation not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    strGetXMLSettings = TCPIPStation.GetXMLSettings
    MsgBox "GetXMLSettings = " & CStr(strGetXMLSettings), vbInformation, "ERROR "

    Set drv = Nothing
    Set station = Nothing
    Set TAPIStation = Nothing
    Set SerialStation = Nothing
End Sub
```

TCPIPStationInterface Properties

BackupServerAddress, TCPIPStationInterface Property

Syntax String = station.BackupServerAddress

Description This property sets or returns the IP address of the Backup Server for the station. Shows the value of the "Backup Server Address" property.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim RASStation As RASStationInterface
    Dim TCPIPStation As TCPIPStationInterface

    Set drv = GetDriverInterface("Modbus TCPIP")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    Set RASStation = station.GetSubStationInterface
    If RASStation Is Nothing Then
        MsgBox "RASStation not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    Set TCPIPStation = RASStation.GetBaseStationInterface
    If TCPIPStation Is Nothing Then
        MsgBox "TCPIPStation not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    TCPIPStation.BackupServerAddress = "192.168.0.10"
    TCPIPStation.SwitchServerTimeout = 20000

    MsgBox "BackupServerAddress = " & TCPIPStation.BackupServerAddress & vbCrLf & _
        "SwitchServerTimeout = " &
        TCPIPStation.SwitchServerTimeout, vbExclamation, "ERROR "
    Set drv = Nothing
    Set station = Nothing
    Set RASStation = Nothing
    Set TCPIPStation = Nothing
End Sub
```

LocalBoundAddress, TCPIPStationInterface Property

Syntax String = station.LocalBoundAddress

Description This property sets or returns the local IP Address that the station will use for the connection. This property is useful when on the PC are present two or more network cards. If the field comes lets voids the IP address of the default card will be used.

Remarks A modification of this value will come acquired only when the socket will open. For example on the project startup before that the communication tasks are executed at least once.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

See the "ServerAddress" property example.

LocalBoundPort, TCPIPStationInterface Property

Syntax long = station.LocalBoundPort

Description This property sets or returns the local Port that the station will use for the connection. If the field comes lets voids the default Port will be used.

Remarks A modification of this value will come acquired only when the socket will open. For example on the project startup before that the communication tasks are executed at least once.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:
See the "ServerAddress" property example.

MaxReceive, TCPIPStationInterface Property

Syntax Long = station.MaxReceive

Description This property sets or returns the Receiving Buffer Queue (Bytes) for the Station.

Remarks A modification of this value will come acquired only when the socket will open. For example on the project startup before that the communication tasks are executed at least once.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim RASStation As RASStationInterface
    Dim TCPIPStation As TCPIPStationInterface
    Dim IMaxReceive As Long
    Dim IMaxSend As Long

    Set drv = GetDriverInterface("Modbus TCPIP")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    Set RASStation = station.GetSubStationInterface
    If RASStation Is Nothing Then
        MsgBox "RASStation not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    Set TCPIPStation = RASStation.GetBaseStationInterface
    If TCPIPStation Is Nothing Then
        MsgBox "TCPIPStation not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    IMaxReceive = TCPIPStation.MaxReceive
    IMaxSend = TCPIPStation.MaxSend

    MsgBox "MaxReceive = " & CStr(IMaxReceive) & vbLf & _
        "MaxSend = " & CStr(IMaxSend), vbInformation, "ERROR "

    Set drv = Nothing
    Set station = Nothing
```

```

        Set RASStation = Nothing
        Set TCPIPStation = Nothing
    End Sub

```

MaxSend, TCPIPStationInterface Property

Syntax Long = station.MaxSend

Description This property sets or returns the Sending Buffer Queue (Bytes) for the Station.

Remarks A modification of this value will come acquired only when the socket will open. For example on the project startup before that the communication tasks are executed at least once.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the "MaxReceive" property example.

RxTimeout, TCPIPStationInterface Property

Syntax Long = station.RxTimeout

Description This property sets or returns the Timeout for the data receiving. This time is in milliseconds.

Remarks A modification of this value will come acquired only when the socket will open. For example on the project startup before that the communication tasks are executed at least once.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim RASStation As RASStationInterface
    Dim TCPIPStation As TCPIPStationInterface
    Dim IRxTimeout As Long
    Dim ITxTimeout As Long

    Set drv = GetDriverInterface("Modbus TCPIP")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    Set RASStation = station.GetSubStationInterface
    If RASStation Is Nothing Then
        MsgBox "RASStation not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    Set TCPIPStation = RASStation.GetBaseStationInterface
    If TCPIPStation Is Nothing Then
        MsgBox "TCPIPStation not found!", vbExclamation, "ERROR "
    End If

```

```

Exit Sub
End If

IRxTimeout = TCPIPStation.RxTimeout
ITxTimeout = TCPIPStation.TxTimeout

MsgBox "RxTimeout = " & CStr(IRxTimeout) & vbCrLf & _
      "TxTimeout = " & CStr(ITxTimeout),vbInformation,"ERROR "

Set drv = Nothing
Set station = Nothing
Set RASStation = Nothing
Set TCPIPStation = Nothing
End Sub

```

ServerAddress, TCPIPStationInterface Property

Syntax String = station.ServerAddress

Description This property sets or returns the Name or the IP Address of the Server to which the station will be connected.

Remarks A modification of this value will come acquired only when the socket will open. For example on the project startup before that the communication tasks are executed at least once.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim RASStation As RASStationInterface
    Dim TCPIPStation As TCPIPStationInterface
    Dim strServerAddress As String
    Dim IServerPort As Long
    Dim strLocalBoundAddress As String
    Dim ILocalBoundPort As Long

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!",vbExclamation,"ERROR "
        Exit Sub
    End If

    Set RASStation = station.GetSubStationInterface
    If RASStation Is Nothing Then
        MsgBox "RASStation not found!",vbExclamation,"ERROR "
        Exit Sub
    End If

    Set TCPIPStation = RASStation.GetBaseStationInterface
    If TCPIPStation Is Nothing Then
        MsgBox "TCPIPStation not found!",vbExclamation,"ERROR "
        Exit Sub
    End If

    strServerAddress = TCPIPStation.ServerAddress
    IServerPort = TCPIPStation.ServerPort
    strLocalBoundAddress = TCPIPStation.LocalBoundAddress
    ILocalBoundPort = TCPIPStation.LocalBoundPort

    MsgBox "ServerAddress = " & CStr(strServerAddress) & vbCrLf & _
          "ServerPort = " & CStr(IServerPort) & vbCrLf & _

```

```

"LocalBoundAddress = " & CStr(strLocalBoundAddress) & vbCrLf & _
"LocalBoundPort = " & CStr(lLocalBoundPort),vbInformation,"ERROR "

Set drv = Nothing
Set station = Nothing
Set RASStation = Nothing
Set TCPIPStation = Nothing
End Sub

```

ServerPort, TCPIPStationInterface Property

Syntax Long = station.ServerPort

Description This property sets or returns the Port number of the Server to which the station will be connected.

Remarks A modification of this value will come acquired only when the socket will open. For example on the project startup before that the communication tasks are executed at least once.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the "ServerAddress" property example.

SwitchServerTimeout, TCPIPStationInterface Property

Syntax Long = station.SwitchServerTimeout

Description This property sets or returns the time, in milliseconds, which passes between the communication with a server error verification and the attempt to connect to another server. Shows the value of the "Switch Server Timeout" property.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the "BackupServerAddress" property example.

TxTimeout, TCPIPStationInterface Property

Syntax Long = station.TxTimeout

Description This property sets or returns the Timeout for the data sending. This time is in milliseconds.

Remarks A modification of this value will come acquired only when the socket will open. For example on the project startup before that the communication tasks are executed at least once.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:
See the "RxTimeout" property example.

1.6.9. RASStationInterface

RASStationInterface Functions

GetBaseStationInterface, RASStationInterface Function

Syntax GetBaseStationInterface

Description This function used in this interface return an object "**TCPIPStationInterface**".

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Object
If Function has been executed successfully it will retrieve an object of type TCPIPStationInterface if , otherwise Nothing is returned.

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim RASStation As RASStationInterface
    Dim SubRASStation As RASStationInterface
    Dim TCPIPStation As TCPIPStationInterface

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!",vbExclamation,"ERROR "
        Exit Sub
    End If

    Set RASStation = station.GetSubStationInterface
    If RASStation Is Nothing Then
        MsgBox "RASStation not found!",vbExclamation,"ERROR "
        Exit Sub
    End If

    Set TCPIPStation = RASStation.GetBaseStationInterface
    If TCPIPStation Is Nothing Then
        MsgBox "TCPIPStation not found!",vbExclamation,"ERROR "
        Exit Sub
    End If

    Set SubRASStation = RASStation.GetSubStationInterface
    If SubRASStation Is Nothing Then
        MsgBox "SubRASStation not found!",vbExclamation,"ERROR "
        Exit Sub
    End If

    Set drv = Nothing
    Set station = Nothing
    Set RASStation = Nothing
```

```

        Set SubRASStation = Nothing
        Set TCPIPStation = Nothing
    End Sub

```

GetSubStationInterface, RASStationInterface Function

Syntax GetSubStationInterface

Description This function used in this interface return an object "**RASStationInterface**".

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Object
 If Function has been executed successfully it will retrieve an object of type RASStationInterface if , otherwise Nothing is returned.

Example:
 See the "GetBaseStationInterface" property example.

GetXMLSettings, RASStationInterface Function

Syntax GetXMLSettings

Description Gets all settings of a station in XML format.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:
 Sub Main
 Dim drv As DriverInterface
 Dim station As StationInterface
 Dim RASStation As RASStationInterface
 Dim strGetXMLSettings As String

 Set drv = GetDriverInterface("Modbus Serial")
 Set station = drv.GetDriverStation("Station1")
 If station Is Nothing Then
 MsgBox "Station not found!",vbExclamation,"ERROR "
 Exit Sub
 End If

 Set RASStation = station.GetSubStationInterface
 If RASStation Is Nothing Then
 MsgBox "RASStation not found!",vbExclamation,"ERROR "
 Exit Sub
 End If

 strGetXMLSettings = RASStation.**GetXMLSettings**
 MsgBox "GetXMLSettings = " & CStr(strGetXMLSettings),vbInformation,"ERROR "

 Set drv = Nothing

```

        Set station = Nothing
        Set RASStation = Nothing
    End Sub

```

IsConnected, RASStationInterface Function

Syntax IsConnected

Description Gets the connection state of the RAS Station. The "True" value indicates that the station is being connected or already connected.

Remarks

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
 True = Station Connected or is being connected
 False = Station not Connected

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim RASStation As RASStationInterface
    Dim bIsConnected As Boolean

    Set drv = GetDriverInterface("Modbus Serial")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    Set TAPIStation = station.GetSubStationInterface
    If TAPIStation Is Nothing Then
        MsgBox "TAPIStation not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    bIsConnected = TAPIStation.IsConnected
    MsgBox "IsConnected = " & CStr(bIsConnected), vbInformation, "ERROR "

    Set drv = Nothing
    Set station = Nothing
    Set TAPIStation = Nothing
End Sub

```

RASStationInterface Properties

DisconnectAfterSecs, RASStationInterface Property

Syntax Long = station.DisconnectAfterSecs

Description This property sets or returns the time in milliseconds after which the modem connection of the RAS Station will hang up. The counter of the time will start from the moment in which all the communication tasks are not in use, and it will be reset if at least one task returns in use before the expiration of the time.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

| | |
|------|------|
| None | None |
|------|------|

Result Long

Example:

See the "PhoneBookEntry" property example.

EnableRASCallOnThisStation, RASStationInterface Property

Syntax Boolean = station.EnableRASCallOnThisStation

Description This property sets or returns the enable to execute the modem connection for the Station.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
True = RAS call enable
False = RAS call disable

Example:

See the "PhoneBookEntry" property example.

EndConnectionTime, RASStationInterface Property

Syntax Date = station.EndConnectionTime

Description This property returns the date and time of end of the last modem connection of the Station.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Date

Example:

See the "**StartConnectionTime**" property example.

LastConnectionTime, RASStationInterface Property

Syntax Date = station.LastConnectionTime

Description This property returns the connection time of the last modem call of the Station.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
|-----------|-------------|

| | |
|------|------|
| None | None |
|------|------|

Result Date

Example:

See the "StartConnectionTime" property example.

LastRASErrorNumber, RASStationInterface Property

Syntax Long = station.LastRASErrorNumber

Description This property returns the last error code occurs during the modem call of the Station.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:

See the "StartConnectionTime" property example.

LastRASErrorString, RASStationInterface Property

Syntax String = station.LastRASErrorString

Description This property returns a description text about the state of the modem call of the Station. This string can be an error message or a state connection message.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

See the "StartConnectionTime" property example.

NumRetries, RASStationInterface Property

Syntax Byte = station.NumRetries

Description This property sets or returns the number of retries to execute for the modem call of the Station when the connection fails.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Byte

Example:

See the "PhoneBookEntry" property example.

Password, RASStationInterface Property

Syntax String = station.Password

Description This property sets or returns the password of the User selected for the RAS connection. This field is not used if the property "PhoneBookEntry" is not null.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

See the "PhoneBookEntry" property example.

PhoneBookEntry, RASStationInterface Property

Syntax String = station.PhoneBookEntry

Description This property sets or returns the name of the connection previously created and configured in the Operating System.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

```
Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim RASStation As RASStationInterface
    Dim strPhoneBook As String
    Dim strPhoneNumber As String
    Dim strUserName As String
    Dim strPassword As String
    Dim nNumRetries As Byte
    Dim IDisconnectAfter As Long
    Dim IRetryAfter As Long
    Dim bEnableRASCall As Boolean
    Dim bPromptForConn As Boolean
    Dim bShowConnDlg As Boolean

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
    Exit Sub
```

```

End If

Set RASStation = station.GetSubStationInterface
If RASStation Is Nothing Then
    MsgBox "RASStation not found!",vbExclamation,"ERROR "
Exit Sub
End If

strPhoneBook = RASStation.PhoneBookEntry
strPhoneNumber = RASStation.PhoneNumber
strUserName = RASStation.UserName
strPassword = RASStation.Password
nNumRetries = RASStation.NumRetries
IDisconnectAfter = RASStation.DisconnectAfterSecs
lRetryAfter = RASStation.RetryAfterSecs
bEnableRASCall = RASStation.EnableRASCallOnThisStation
bPromptForConn = RASStation.PromptForConnection
bShowConnDlg = RASStation.ShowConnectionDlg

MsgBox "PhoneBookEntry = " & CStr(strPhoneBook) & vbLf & _
"PhoneNumber = " & CStr(strPhoneNumber) & vbLf & _
"UserName = " & CStr(strUserName) & vbLf & _
>Password = " & CStr(strPassword) & vbLf & _
"NumRetries = " & CStr(nNumRetries) & vbLf & _
"DisconnectAfterSecs = " & CStr(IDisconnectAfter) & vbLf & _
"RetryAfterSecs = " & CStr(lRetryAfter) & vbLf & _
"EnableRASCallOnThisStation = " & CStr(bEnableRASCall) & vbLf & _
"PromptForConnection = " & CStr(bPromptForConn) & vbLf & _
>ShowConnectionDlg = " & CStr(bShowConnDlg),vbInformation,"ERROR "

Set drv = Nothing
Set station = Nothing
Set RASStation = Nothing
End Sub

```

PhoneNumber, RASStationInterface Property

Syntax String = station.PhoneNumber

Description This property sets or returns the telephone number to call. This field is not used if the property "PhoneBookEntry" is not null.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

See the "PhoneBookEntry" property example.

PromptForConnection, RASStationInterface Property

Syntax Boolean = station.PromptForConnection

Description This property allows you to show or hide the connection confirmation dialog window. If this property is true, before start connection, the Supervisor will show a confirmation dialog window. Only if the user will click the OK button the Supervisor will start the connection.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
True = show the dialog window before connecting
False = don't show the dialog window and start connection

Example:
See the "PhoneBookEntry" property example.

RetryAfterSecs, RASStationInterface Property

Syntax Long = station.RetryAfterSecs

Description This property sets or returns the time in milliseconds after which the modem connection will be retried if the previous call has failed.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Long

Example:
See the "PhoneBookEntry" property example.

ShowConnectionDlg, RASStationInterface Property

Syntax Boolean = station.ShowConnectionDlg

Description This property allows you to show or hide the connection status dialog window.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Boolean
True = show dialog window
False = hide dialog window

Example:
See the "PhoneBookEntry" property example.

StartConnectionTime, RASStationInterface Property

Syntax Date = station.StartConnectionTime

Description This property returns the date and time of begin of the last modem connection of the Station.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Date

Example:

```

Sub Main
    Dim drv As DriverInterface
    Dim station As StationInterface
    Dim RASStation As RASStationInterface
    Dim dStartConn As Date
    Dim dEndConn As Date
    Dim dTotalConn As Date
    Dim dLastConn As Date
    Dim lLastError As Long
    Dim strLastErrorString As String

    Set drv = GetDriverInterface("Modbus TCP/IP")
    Set station = drv.GetDriverStation("Station1")
    If station Is Nothing Then
        MsgBox "Station not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    Set RASStation = station.GetSubStationInterface
    If RASStation Is Nothing Then
        MsgBox "RASStation not found!", vbExclamation, "ERROR "
        Exit Sub
    End If

    dStartConn = RASStation.StartConnectionTime
    dEndConn = RASStation.EndConnectionTime
    dTotalConn = RASStation.TotalConnectionTime
    dLastConn = RASStation.LastConnectionTime
    lLastError = RASStation.LastRASErrorNumber
    strLastErrorString = RASStation.LastRASErrorString

    MsgBox "StartConnectionTime = " & CStr(dStartConn) & vbCrLf & _
        "EndConnectionTime = " & CStr(dEndConn) & vbCrLf & _
        "TotalConnectionTime = " & CStr(dTotalConn) & vbCrLf & _
        "LastConnectionTime = " & CStr(dLastConn) & vbCrLf & _
        "LastTAPIError = " & CStr(lLastError) & vbCrLf & _
        "LastTAPIErrorString = " & CStr(strLastErrorString), vbInformation, "ERROR "

    Set drv = Nothing
    Set station = Nothing
    Set RASStation = Nothing
End Sub

```

TotalConnectionTime, RASStationInterface Property

Syntax Date = station.TotalConnectionTime

Description This property returns the total connection time of the modem call of the Station. The time is the sum of all calls executed from the start of the Project.

Remarks This property is read-only.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result Date

Example:

See the "**StartConnectionTime**" property example.

UserName, RASStationInterface Property

Syntax String = station.UserName

Description This property sets or returns the name of the User selected for the RAS connection. This User must be authenticated by the Server. This field is not used if the property "PhoneBookEntry" is not null.

Remarks A modification of this value will come acquired only to the following call.

| Parameter | Description |
|-----------|-------------|
| None | None |

Result String

Example:

See the "PhoneBookEntry" property example.

1.7. Errors

Errors are grouped by scope and are listed in alphabetical order.

| Design errors | | | | |
|--|--------------|----------------|---|---|
| Error message | Scope | Gravity | Possible cause | Possible solution |
| Internal Error in file '<file_name>' at the line <line_number>. Please contact the Technical Support | design | serious | problem in user interface controls creation or configuration files access | call support center |
| Invalid Address Offset Variable. Enter the name of an existing integer variable | design | warning | the selected variable is not of a valid type. | choose an existing integer variable from the tags database. |
| Invalid conditional variable '<variable_name>' for the task '<task_name>' | design | warning | the selected variable is not valid: does not exist or has been cancelled or is of a wrong data type. | choose an existing integer variable from the tags database. |
| Invalid conditional variable. Enter only one existing variable of numeric type. | runtime | warning | the conditional variable is not valid: does not exist or has been cancelled or is of a wrong data type. | choose an existing integer variable of the tags database. |
| ListView out of memory | design | serious | there is not enough memory to manage operation on stations list or tasks list. | free some memory closing processes or applications, restart Supervisor or PC if needed. |
| Name <name>' is already in use. Please, choose a different name | design | warning | station name or task name already existing. Duplicated are not allowed. | change station or task name |
| Task size exceeds the maximum allowed size for this protocol | runtime | warning | the total size of the variables belonging to the task is bigger than the maximum allowed size for this protocol | reduce the number of variables associated to this task |
| The selected Dynamic Settings String is either invalid or not allowed for this Variable! | design | warning | the dynamic address string specified for the variable is invalid or not allowed | check address format and variable data type |

| | | | | |
|--|---------|---------|--|--|
| The selected COM port is either not supported or is being used by another application. Please select another port. | design | warning | the port number set in Station/Serial port settings/Port property is invalid | allowed values for Station/SerialPort settings/Port property are positive integer (not zero) |
| Value cannot be empty! Please enter a value | design | warning | ethernet drivers only. The values for TCPIP station/Server address property or Server port property are invalid. | enter a valid Server address: IP address in xxx.xxx.xxx.xxx format, or server name i.e. 'server1' enter a valid Server port number :positive integer (not zero) |
| Value cannot be zero! Please enter a number greater than 0 | design | warning | the value for Rx queue size or Tx queue size in Station/Queue size properties is invalid | enter a positive integer (not zero) |
| Variable '<variable_name>' is not valid for the task '<task_name>' | runtime | warning | the selected variable is not valid: does not exist or has been cancelled or is of a wrong data type | choose an existing variable of the tags database. Check task design and addressing rules. |

Address validation errors

| Error message | Scope | Gravity | Possible cause | Possible solution |
|---|---------|---------|--|---|
| Job <job_name> (station <station_name>) is invalid and cannot be promoted to the protocol mng | runtime | serious | the task is not valid and will not be executed | check task communication parameters |
| Setting of addresses for fields of the structure <structure_name> interrupted on unsupported field <field_name> | runtime | warning | error in managing structure members' addressing due to member type (string or double): only structure members preceding them in structure's members order can be properly addressed. | modify structure members or use single tags for data type like string or double |
| The Task <task_name> for the Station <station_name> is invalid | runtime | serious | the task is not valid and will not be executed | check task communication parameters |

Hardware errors

| Error message | Scope | Gravity | Possible cause | Possible solution |
|-------------------------------|---------|---------|--|--|
| Error loading <drivename>.dll | runtime | fatal | the driver dll file has not been found | check if <drivename>.dll file is present in 'Drivers' subfolder under Supervisor installation folder |

Communication errors

| Error message | Scope | Gravity | Possible cause | Possible solution |
|--|--------------------|---------|--|--|
| Communication error : station <station_name>, error <error_description> | runtime | various | this is the generic message for communication errors: error description gives a detail explanation of the error source | check the possible error source as suggested by error description |
| Error ! Driver has already been initialized ! | runtime | fatal | unexpected behavior of the driver, an attempt to initialize it was made but the driver was already running. | call support center |
| Failed to open the communication device. The configured COM port is either not supported or is being used by another application | communication test | serious | failed to open serial port (serial drivers) or communication socket | check if port number is correct, check cables connection, check if port is in use by another application, check device accessibility |

Communication messages

| Message | Scope | Gravity | Possible cause | Possible solution |
|--|---------|---------|---|-------------------|
| Communication established : station <station_name> | runtime | info | the communication has been established properly | |

| | | | | |
|--|------------------------|------|--|--|
| Communication restored | runtime | info | the communication has been restored after a communication problem | |
| Station quality changed : station <station_name>, quality bitfield <quality_value> | runtime (debug window) | info | the quality of the station's data changed: possible values are bad, good, uncertain, not connected | |
| Task <task_name> (Station <station_name>) is now in use... | runtime (debug window) | info | if in-use management is on, the dynamic tasks are activated or deactivated according to tags in use. | |
| Task <task_name> (Station <station_name>) is now NOT in use... | runtime (debug window) | info | if in-use management is on, the dynamic tasks are activated or deactivated according to tags in use. | |
| The station <station_name> has switched from server <primary_server_name> to <secondary_server_name> | runtime | info | applies to Ethernet driver only. If a backup server has been defined in Network services/Redundancy property and the primary server is down, the station connects to the backup server, or vice-versa. | |

| Serial communication errors (serial drivers only) | | | | |
|--|---------|---------|---|---|
| Error message | Scope | Gravity | Possible cause | Possible solution |
| Break Detect ! | runtime | serious | the line state is a break state, so there is no communication | check cable connection and device settings on both device and supervision |
| Carrier Detect Timeout ! | runtime | serious | the driver did not receive the expected answer in the maximum allowed interval (timeout) | check cable connection and device settings on both device and supervision |
| Clear To Send Timeout ! | runtime | serious | the driver did not receive the expected answer in the maximum allowed interval (timeout) | check cable connection and device settings on both device and supervision |
| Data Set Ready Timeout ! | runtime | serious | the driver did not receive the expected answer in the maximum allowed interval (timeout) | check cable connection and device settings on both device and supervision |
| FATAL ERROR! The Windows message queue is full! Serial Comm messages lost! Increase the message queue depth. | runtime | fatal | self explaining | self explaining |
| Frame Error ! | runtime | serious | noise in the communication line | remove possible noise sources, check parity bit or stop bit values |
| Overrun Error ! | runtime | serious | incoming data detected while previous data have still not been received; the operating system is too busy | reduce baud rate value, check hardware, close applications that can catch operating system resources |
| Parity Error ! | runtime | serious | noise on the communication line | remove possible noise sources, check parity bit value |
| Rx Timeout ! | runtime | serious | the driver did not receive the expected answer in the maximum time interval allowed (timeout) | check cable connection and device settings on both device and supervision, check RS232/RS485 converter settings, if any |
| Rx Buffer Overflow ! | runtime | serious | the reception buffer is full, possible loss of data | increase Station/Queue size/Rx queue size, activate Station/serial port settings/Flow control property or hardware flow |

| | | | | |
|--|---------|---------|--|---|
| | | | | control if supported by the device |
| The selected COM port is either not supported or is being used by another application. Please select another port. | runtime | serious | the COM port does not exist or is already used by another application. | check port number in Station/SerialPort settings/Port property; free the port closing the application using it and restart the PC if needed |
| Tx Buffer Full ! | runtime | serious | the transmission buffer is full, messages are not exiting | increase Station/Queue size/Tx queue size |
| Unexpected WM_QUIT received ! | runtime | info | the driver received a close message from Windows and is shutting down | |

| TAPI errors (serial drivers only) | | | | |
|--|--------------|----------------|---|--|
| Error message | Scope | Gravity | Possible cause | Possible solution |
| Call to <phone_number> has been hung up | runtime | info | the communication has been stopped by the caller | |
| Disconnected: Bad Address | runtime | serious | the destination address is invalid | check the if phone number to call is correct |
| Disconnected: Busy | runtime | serious | the remote user's station is busy | wait for the remote user's station to be free |
| Disconnected: Congestion | runtime | serious | the network is congested, so the call can not be placed | wait for the line to be free |
| Disconnected: Incompatible | runtime | serious | the remote user's station equipment is incompatible with the type of call requested | check remote user's station equipment |
| Disconnected: Local phone picked up | runtime | serious | the call was picked up from elsewhere | |
| Disconnected: No Answer | runtime | serious | the call has been placed but there was no answer, so it has been disconnected | |
| Disconnected: No Dial Tone | runtime | serious | a dial tone was not detected when expected | check cables and line |
| Disconnected: Unknown reason | runtime | serious | the call has been disconnected for unknown reasons | |
| Disconnected: Unreachable | runtime | serious | the called number can not be reached | |
| Line busy | runtime | warning | trying to place a call while the line is already busy | wait for the line to be free |
| Placing call to <phone_number> ... | runtime | info | specifies the phone number is going to be called | |
| Remote Party Disconnected | runtime | serious | the communication has been stopped by the remote party (the receiver) | |
| Remote Party rejected call | runtime | serious | the call has been rejected by the remote party (the receiver) | |
| TAPI in special information state, probably couldn't dial number | runtime | serious | the connection has failed | |
| Unable to find modem associated with <port> | runtime | serious | modem has not been found on the specified port | check in the control panel if the right port is assigned to the modem, check modem settings and cables |

| Bridging errors (serial drivers only) | | | | |
|---|--------------|----------------|--------------------------------|--------------------------|
| Error message | Scope | Gravity | Possible cause | Possible solution |
| A modem has been connected to the Bridging Port, station <station_name> | runtime | info | | |
| A remote device or computer has taken control through the Bridging Port! - station <station_name> | runtime | info | the bridging operation started | |

| | | | | |
|--|-------------------------------|---------|--|---|
| Connection window creation failed : station <station_name> | runtime or communication test | serious | internal error: the driver application can not create the dialog window, for example due to a lack of memory resources | check available memory resources |
| Could not initialise the Modem on the Bridging Port, station <station_name> | runtime | serious | the command for initializing modem has been sent but there was no proper answer | check modem state, settings and cables |
| Could not perform read operation on the Bridging Port, station <station_name> | runtime | serious | can not read data | check modem state, settings and cables |
| Could not perform write operation on the Bridging Port, station <station_name> | runtime | serious | can not write data | check modem state, settings and cables |
| Device control has been restored - Bridging ended! - station | runtime | info | end of the bridging operations | |
| Disconnection sequence received on the Bridging Port, station | runtime | info | there is no more bridging activity so the communication has been stopped | |
| The modem has been disconnected from the Bridging Port, station <station_name> | runtime | info | self explanatory | |
| Too many characters received on the Bridging Port, station | runtime | serious | too many characters have been received with respect to the internal buffer size. | reset the bridging closing the modem port |

RAS errors (ethernet drivers only)

| Error message | Scope | Gravity | Possible cause | Possible solution |
|--|-------------------------------|---------|---|---|
| Connection window creation failed : station <station_name> | runtime or communication test | serious | internal error: the driver application can not create the dialog window, for example due to a lack of memory resources | check available memory resources |
| RAS Dial Error : <error-description>, Station <station_name> | runtime or communication test | serious | see error description | see error description |
| Station <station_name>' is about to dial-up. Continue ? | runtime | info | ask for confirmation before calling. This message appears only if 'Prompt before connect' property in station's RAS settings has been set to True. | |
| Undefined RAS Dial Error <error_number>, Station<station_name> | runtime | serious | generic error on RAS connection | check RAS parameters in driver station properties, check cable connection, check RAS settings in RAS server (PC receiving call) |

| Basic Script Interface | | | | |
|---|---------|---------|--|--|
| Error Message | Scope | Gravity | Possible cause | Possible solution |
| Closing driver ... | runtime | info | This message displayed when the DriverInterface.CloseDriver function is invoked. | |
| Driver closed | runtime | info | The DriverInterface.CloseDriver function has completed with success. | |
| Error ! Driver has already been initialized ! | runtime | warning | The DriverInterface.OpenDriver function has already been initialized. | If you wish to re-initialize driver, call the DriverInterface. |

| | | | | |
|--|---------|---------|---|---|
| | | | | CloseDriver before the DriverInterface. OpenDriver |
| Error! Driver already closed or application is shutting down! | runtime | warning | The DriverInterface.CloseDriv er has been called with a inactive driver. | |

1.8. About

Through this window you can checkout the selected **Driver version** and any associated descriptions and comments.



Always check whether the Supervisor Driver file is the latest available, otherwise it would be in your best interests to update the file (.DLL) by downloading it from the local dealer support web site according to the modalities provided.



Movicon™ is a trademark of Progea, related to the HMI/SCADA platform entirely developed and produced by Progea. © 2016 All Rights reserved.

No part of this document or of the program may be reproduced or transmitted in any form without the express written permission of Progea.

Information in this document is subject to change without notice and is not binding in any way for the company producing it.



Via D'annunzio 295
41123 Modena - Italy
Tel. +39 059 451060
Fax +39 059 451061
Email: info@progea.com
Http://www.progea.com



Via XX Settembre, 30
Tecnocity Alto Milanese
20025 Legnano (MI) Italy
Tel. +39 0331 486653
Fax +39 0331 455179
Email: willems@progea.com



Progea Deutschland GmbH
Marie-Curie-Str. 12
D-78048 VS-Villingen
Tel: +49 (0) 7721 / 99 25 992
Fax: +49 (0) 7721 / 99 25 993
info@progea.de



Progea International Ltd
via Sottobisio 28
6828 Balerna - Switzerland
tel +41 (91) 9676610
fax +41 (91) 9676611
international@progea.com



Progea North America Corp
2380 State Road 44 suite C
Oshkosh, WI 54904
Tel. +1 (888) 305 2999
Fax. +1 (920) 257 4213
info@progea.us