I/O modules configuration

EX series

update: 16-06-2017

Table of contents

1. I/O modules with Modbus protocol

The interface modules EX series are particularly suitable for use with the line of *OVERDIGIT* compact PLCs, but their features allow the application in many other areas. The standard communication Modbus protocol makes these modules extremely versatile because are directly manageable by many commercial systems such as PLC, PC and other control devices.

Compliance with the Modbus standard and configurability of the communication parameters, readily enables integration of the modules in many applications. However, to achieve higher performance respect what normally is achieved with Modbus RTU systems, some custom functions, as allowed by the protocol, have been implemented. These functions are designed to minimize the update time of I/O resources in particular the performances of the modules have been further optimized for the *OVERDIGIT* PLCs.

The implementation of a single custom command code for the complete configuration of communication parameters simplifies this operation and allows the values settings with a wide range of variation. In particular, the Baudrate is selectable from 300b/s up to a maximum of 1Mb/s providing the value directly in b/s.

Three other custom command codes allow, in a single exchange of frames, the full update of the entire area for Input, Output or both. In this way the overhead communication time is reduced drastically respect to the use of multiple standard function codes for reading and writing all the I/O.

For *OVERDIGIT* PLCs also other optimizations of the communication are implemented, particulary during the frames exchange in the case of high Baudrate (above 100kb/s). CoDeSys libraries and module resources description files are provided to allow a direct and simple I/O updating based on the informations derived from the "PLC Configuration" of development environment.

1.1. Communication parameters

The Modbus protocol implemented in the EX series modules can be configured using the value of some parameters permanently stored in the internal memory of the device.
The parameters for the configuration are shown in the following table:

Parameter	Range	Default	Description
Address	1 ÷ 247	1	Slave address on network
Baudrate	300 ÷ 1000000	9600	Communication speed (units bit/s)
Parity	0=No (2 stop bits) 1=Odd 2=Even 3=No (1 stop bit)	Even	Type of parity for 8 data bits
Reset time	0 ÷ 60000 (0=No)	No	Maximum time (units 10ms) for outputs reset
Replay delay	0 ÷ 100 (0=No)	No	Delay (units 100µs) before replay sending

NOTE: To restore the default configuration the power supply to the module should be applied with the PG button held down at least about 3".

"Address" uniquely identifies each slave module connected on the RS485 multi-point network, allowing the sending of a protocol command only to a specific node.

"Baudrate" sets the transmission speed of the serial bits on the network. The choice of this value is very important and must take into account several aspects. A high transmission speed reduces the total communication time making the system more powerful but also requires connection lines shorter and made with specific cables and precautions. A low speed instead allows to reach high distances, also in the order of hundreds of meters, but the time necessary for the data exchange increases accordingly.

"Parity" selects the type of control that was added within each byte transmitted. This involves the use of a 9th bit to verify the presence of possible errors on an odd amount of the 8 data bits. According to the official Modbus specification the "No" parity mode involves the use of 2 stop bits but this is also available with only one stop bit.

"Reset time" sets the maximum time allowed between two successive and correct sequences of communication. In case of absence or abnormality of the frames, the watchdog timer inside the module will automatically turn off its outputs.

"Replay delay" sets the waiting time before the module responds to the master after receiving a command. This delay reduces the performance of communication whereby, using the module with *OVERDIGIT* PLCs, must be left disabled.

1.2. Changing the parameters

The modules are supplied with the default configuration already installed whereby they are directly usable only in the case of a single module (address 1) and with particular values of the parameters. The change of the communication parameters takes place via the sending of specific Modbus protocol commands on the serial bus.
Of course to send a configuration command **known values of the parameters**, currently in use for communication, are necessary.
Furthermore it’s necessary to communicate only with **a module at a time** between those connected on the network, so that a different slave address for each specific module can be set. This can be obtained by supplying, or by connecting to the RS485 network, or by enabling in program mode only one module at a time so that the configuration command is accepted by a single device.

Pressing the **PG button** during the module power-on, the default values according to the above table are programmed. This is therefore a possible known condition of communication with which to send the command for a new configuration.
NOTE: After sending the command to write the configuration, the module power supply must be restarted so that the communication is initialized with the new stored values.

Alternatively, to obtain a configuration of known communication, the PG button, with module already powered, can be used. Pressing PG button for about 3", with power already present, the module enters a temporary state of communication, indicated by a flashing green LED, using the following parameters:

Parameter	Value
Address	248
Baudrate	9600
Parity	Even

Pressing PG button again, this temporary condition ends and communication resumes with the previously saved settings. To permanently change the values, the following steps must be followed:

1) With module turned on, press the PG button for 3" (until the green LED flashes)
   → The module resets the communication with the temporary parameters

2) Send on serial bus the Modbus command to write the configuration
   → The module writes the new values to the permanent memory

3) Press the PG button again
   → The module reinitializes communication with the new stored values

NOTE: When the temporary parameters state is ended by PG button pressing, communication is automatically re-initialized with the stored values without the need of module rebooting.

The procedure for configuring the communication parameters must be performed for each module connected to the network using a different slave address.
To send the configuration command can be used the following tools:

1) Modbus-Tool software for configuration via PC
2) Applications and CoDeSys examples for configuration via PLC
3) Development of a custom application for configuration

The first two tools are explained in the following chapters of this manual. Below there are some information about the Modbus commands used to configure communication.
The direct use of these commands allows the realization of specific configuration tools with different development environments, or the integration of configuration functionality into other applications.
These commands are also used in the first two solutions listed but, for these, it's not necessary to know how they work because the tools are ready to use.

1.3. Standard commands for the configuration

The configuration parameters are stored permanently in the module and associated with a specific area of "Holding Registers" data block defined by the Modbus protocol:

Holding Register	Parameter	Function Codes	Description
9000	Address	3, 6, 16, 23	Slave address on network (1 ÷ 247)
9001	Baudrate	3, 6, 16, 23	Communication speed (units 100bit/s)
9002	Parity	3, 6, 16, 23	Type of parity for 8 data bits
9003	Reset time	3, 6, 16, 23	Maximum time (units 10ms) for outputs reset
9004	Replay delay	3, 6, 16, 23	Delay (units 100µs) before replay sending

Each parameter corresponds to a register of word type and the range of variation allowed is shown in the above table. For the Baudrate parameter must be considered an operation of rescaling the value by a factor 100 to contain the maximum value of 1Mb/s within a word variable. For this reason, in the case of configuration using standard commands, the values used for the Baudrate parameter will be multiples of 100.
NOTE: With the writing of one or more communication parameters the value in the permanent memory is changed, but only on a module restart, the communication will be reinitialized with the new stored values.

1.4. Custom commands for the configuration

To set a different configuration on module permanent memory, also the **custom function code 110** can be used. The data area of the command must be an array of bytes (with a total of 8) containing the different types of data needed according to the following format:

Offset	Parameter	Data type
0	Address	byte
1 ÷ 4	Baudrate	double word
5	Parity	byte
6 ÷ 7	Reset time	word

The structure of the complete write command for configuration is therefore as follows:

Address (1 byte)	Code 110 (1 byte)	Writing data area (8 bytes)	Checksum (Low byte)	Checksum (High byte)

After writing values into the permanent memory, the module responds with an echo of the command but without the data area:

Address (1 byte)	Code 110 (1 byte)	Checksum (Low byte)	Checksum (High byte)

The current configuration can be read using the **custom function code 109** (without data):

Address (1 byte)	Code 109 (1 byte)	Checksum (Low byte)	Checksum (High byte)

In response, the module provides a frame similar to that used in the transmission of the write command:

Address (1 byte)	Code 109 (1 byte)	Reading data area (8 bytes)	Checksum (Low byte)	Checksum (High byte)

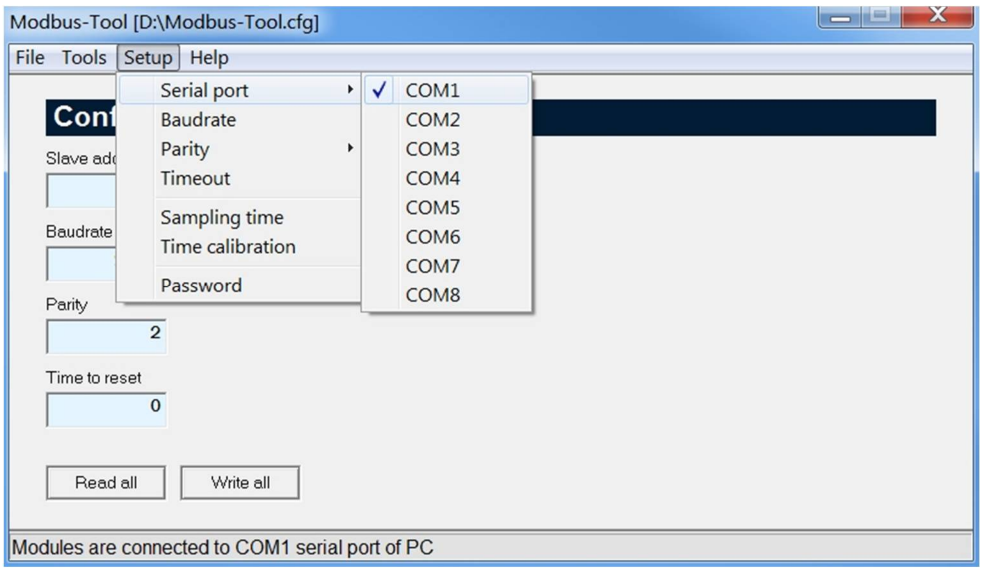
As required by the Modbus standard, data is represented in Big-Endian notation. This implies that, for the values of dimension greater than a byte, the most significant byte is transmitted first. However, this notation is not used for the checksum field, added at the end of the frame, since in this case the least significant byte is transmitted first.
For detailed information on the Modbus protocol the reference document of this standard is available on http://www.modbus.org website.

2. Modbus-Tool utility program

The **Modbus-Tool** software for PC is used for all operations of configuration and I/O testing on the EX series modules.
The tool allows the reading and writing of the Modbus configuration of a specific slave module after it has been placed in temporary communication mode (Address 248, Baudrate 9600, Even parity) using the PG button on the powered module.
Other functionalities are possible such as a manual test of the single I/O resources of a selected slave on the network and manual testing of single function codes of the protocol, acting as Modbus master simulator. In addition, the tool includes a powerful 32 channels analyzer to acquire and modify the values of the Modbus registers on more slaves. The values can be displayed as a trend graph and stored in a logging file.

2.1. Installation and setup of the program

To install the Modbus-Tool software the Modbus-Tool_ Vxxx_Setup.exe program must be executed, following the onscreen instructions. After installation, by running the tool, the main screen of the configuration mode will be presented:



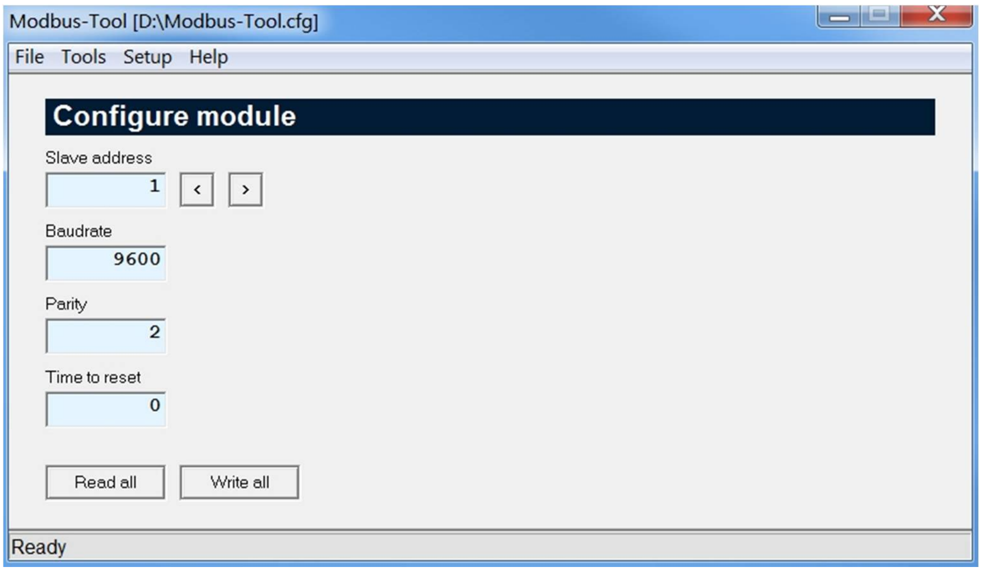
Modules are connected to COM1 serial port of PC

Before using the software, the PC serial communication port, baudrate, parity and timeout of connection must be configured by the **"Setup"** menu.
NOTE: For the functions performed by the tool, a RS485 serial interface must be installed on the PC, using a commercial RS232/RS485 or USB/RS485 adapter.

2.2. “Configure module” menu

The tool provides four modes of use selectable by the **"Tools"** menu.
The first mode is the "Configure module" that allows the parameters configuration of the module activated in programming. For this operation the PG button, placed on the front of the module, must be pressed for 3" to temporarily activate the known communication parameters used for the configuration.

At this point, the module is ready to receive a new configuration. After setting the chosen values in the boxes, writing must be confirmed with the **"Write all"** button:

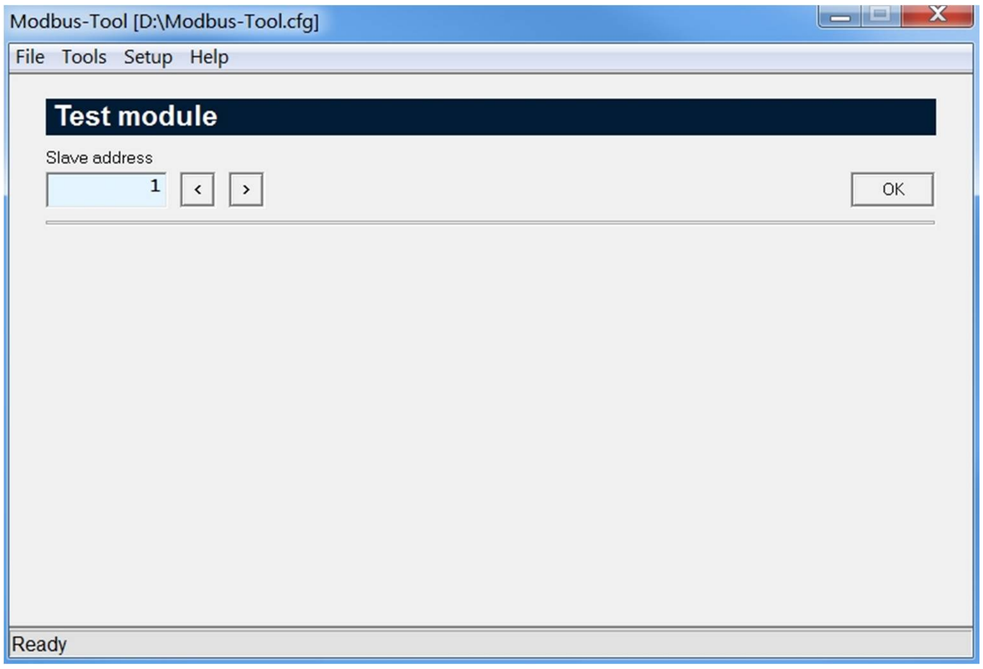


Slave address: 1
Baudrate: 9600
Parity: 2
Time to reset: 0

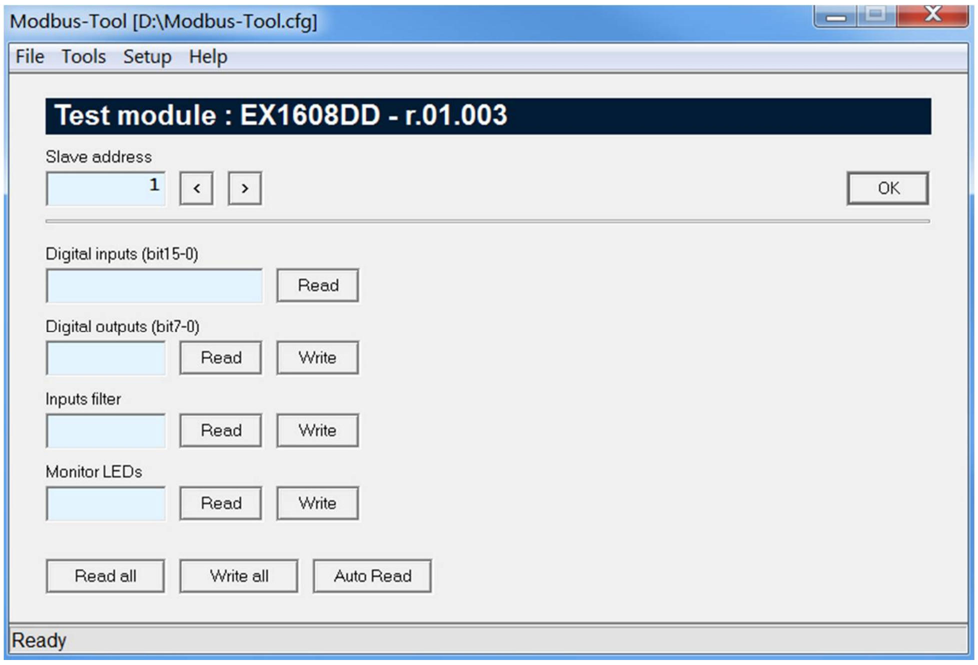
To read the configuration currently stored in the module the **"Read all"** button can be used. The reading values will be inserted in the appropriate box.
When the operations of writing and reading the configuration are completed, return the slave module to the state of normal operation by pressing the PG button again.
NOTE: The configuration values transferred to the module are stored in the "Configure module" tool and are independent from the communication parameters selected in the "Setup" menu, used by the features of the other tools.

2.3. “Test module” menu

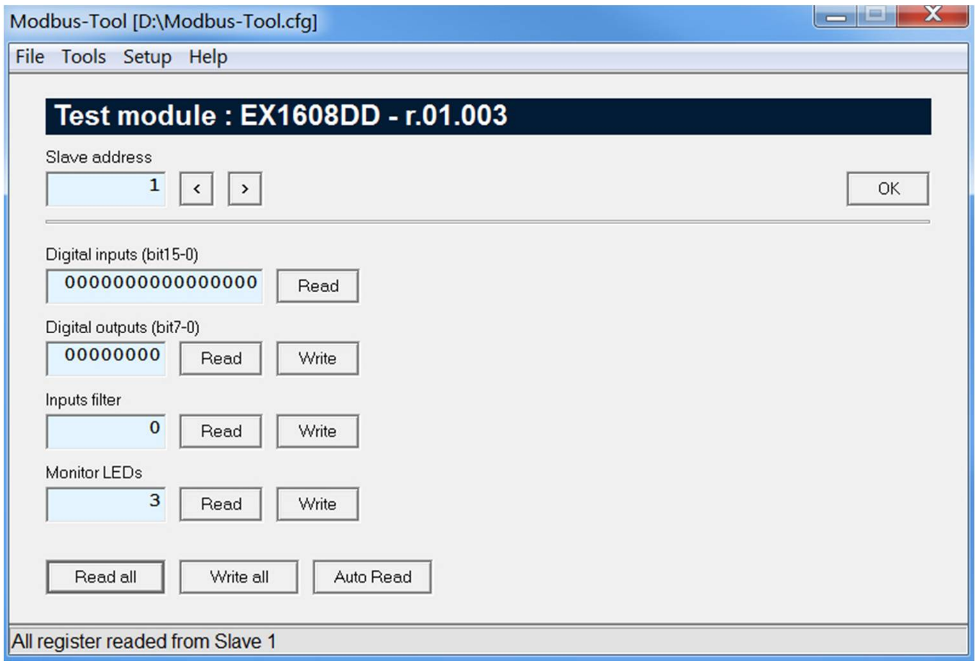
The second mode of tool operation allows to test manually all the I/O resources of a slave at a given address. For communication on serial bus the parameters currently selected in the "Setup" menu are used, while the slave address must be set directly in this screen:



After selecting the slave address of the module to be tested, confirm with the **"OK"** button.
If the module is present on the network, its identification is performed using function code 17 of protocol:



The module is now ready to receive all the commands for reading and writing on the addresses of the available I/O resources. This screen will be different according on the module found at the chosen address and the layout will be adapted to the specific resources provided by the device.
On the right of the boxes of each resource, buttons for reading and/or writing individual values are available, while below "Read all" and "Write all" buttons allow the simultaneous reading and writing of the values of all resources.
Using the **"Read all"** button the reading values will be presented in the appropriate boxes:

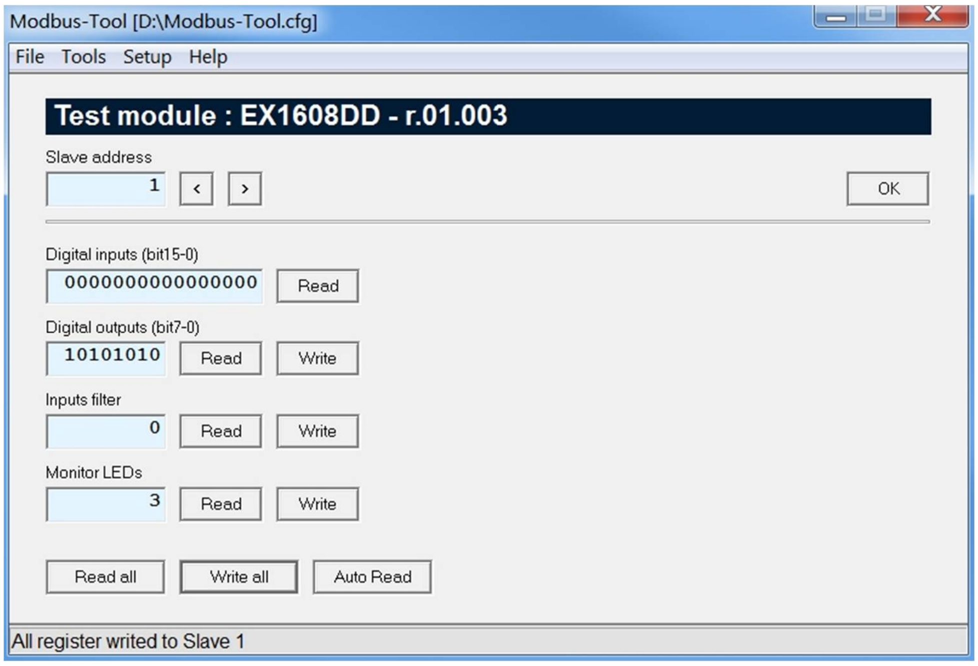


Test module : EX1608DD - r.01.003

Slave address: 1
Digital inputs (bit15-0): 0000000000000000
Digital outputs (bit7-0): 00000000
Inputs filter: 0
Monitor LEDs: 3

All register readed from Slave 1

After entering the values in the boxes of the output resources of module, the **"Write all"** button executes their contemporary writing into the slave:

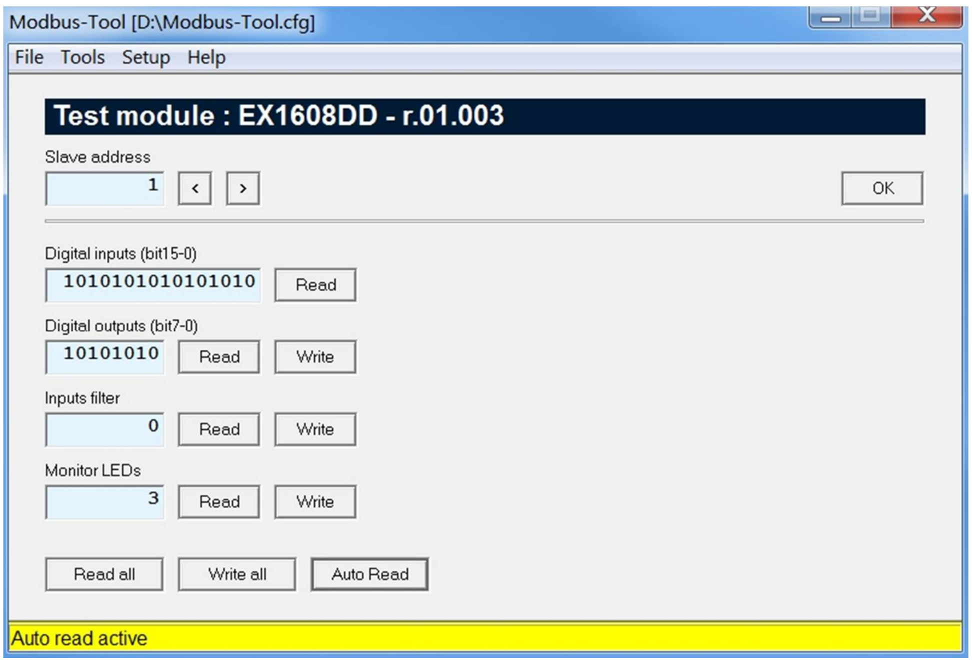


Test module : EX1608DD - r.01.003

Slave address: 1
Digital inputs (bit15-0): 0000000000000000
Digital outputs (bit7-0): 10101010
Inputs filter: 0
Monitor LEDs: 3

All register writed to Slave 1

The **"Auto Read"** button enables the continuous reading of all resources with a repetition period set in the **"Sampling time"** parameter of the "Setup" menu:



Test module : EX1608DD - r.01.003

Slave address: 1

Digital inputs (bit15-0): 1010101010101010

Digital outputs (bit7-0): 10101010

Inputs filter: 0

Monitor LEDs: 3

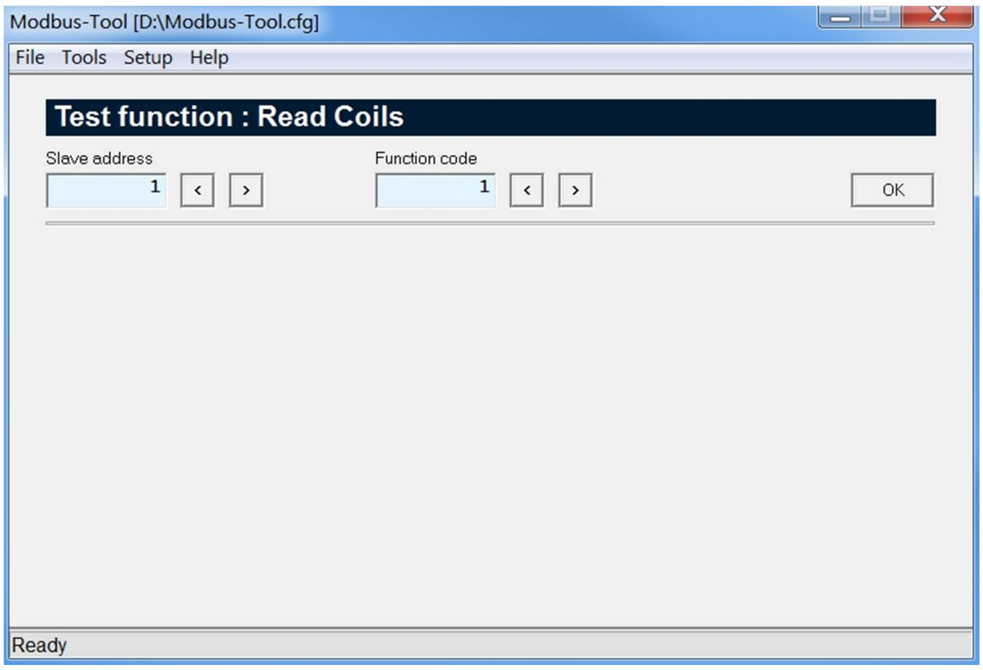
Auto read active

Any small correction of the sampling time can be adjusted with the **"Time calibration"** parameter.

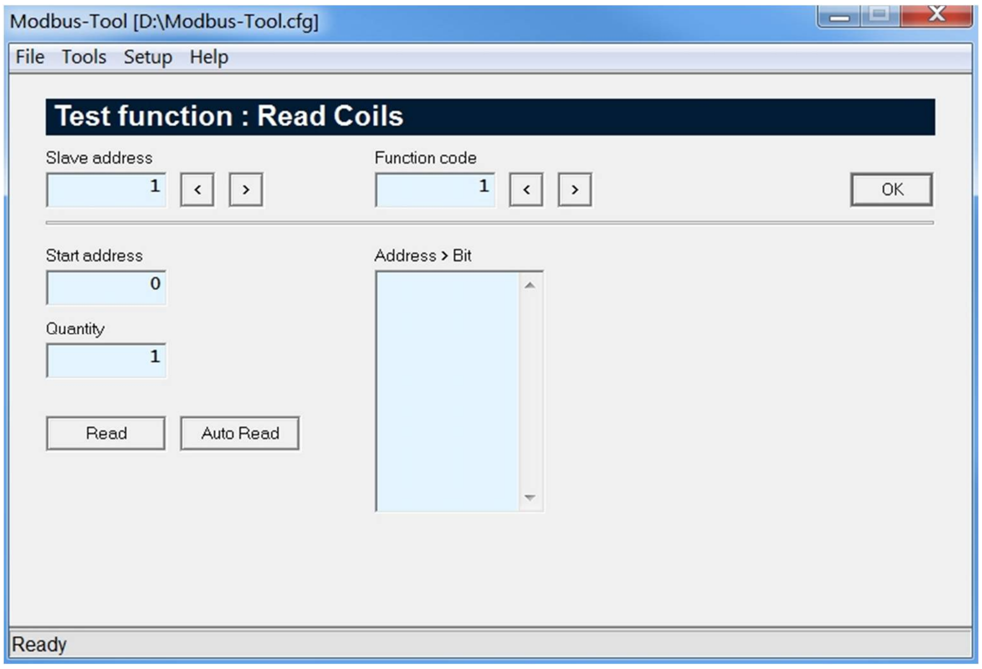
Selecting a different slave address, the screen of the current module will be deleted.
NOTE: Only after confirming with "OK" button, the communication for auto-recognition of module, with the selected slave address, will be executed.

2.4. “Test function” menu

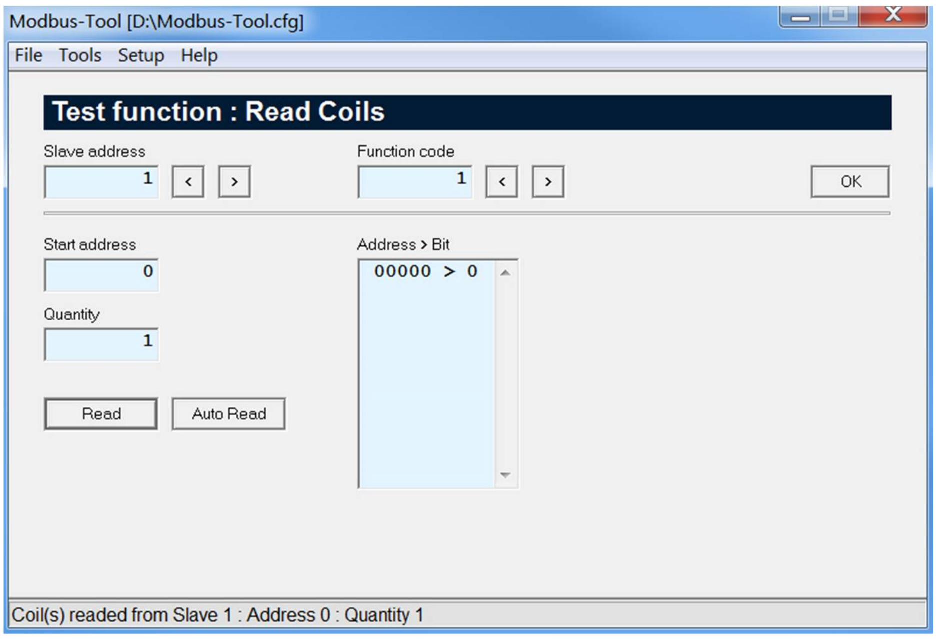
The third mode of tool using allows the testing of individual function codes of protocol on a slave at a given address. The last command code used is visualized as in this example:



After selecting the slave address and function code required, confirm with the **"OK"** button. The new layout of the window depends on the specific and selected command. For example, by selecting the function code 1, this layout is obtained:



The command code 1 allows the reading of the value of one or more bits of memory (Coils). For this purpose, the start address and the number of bits must be specified before activating the reading with the **"Read"** button. The received result is displayed in the box on the right:



Test function : Read Coils

Slave address: 1

Function code: 1

Start address: 0

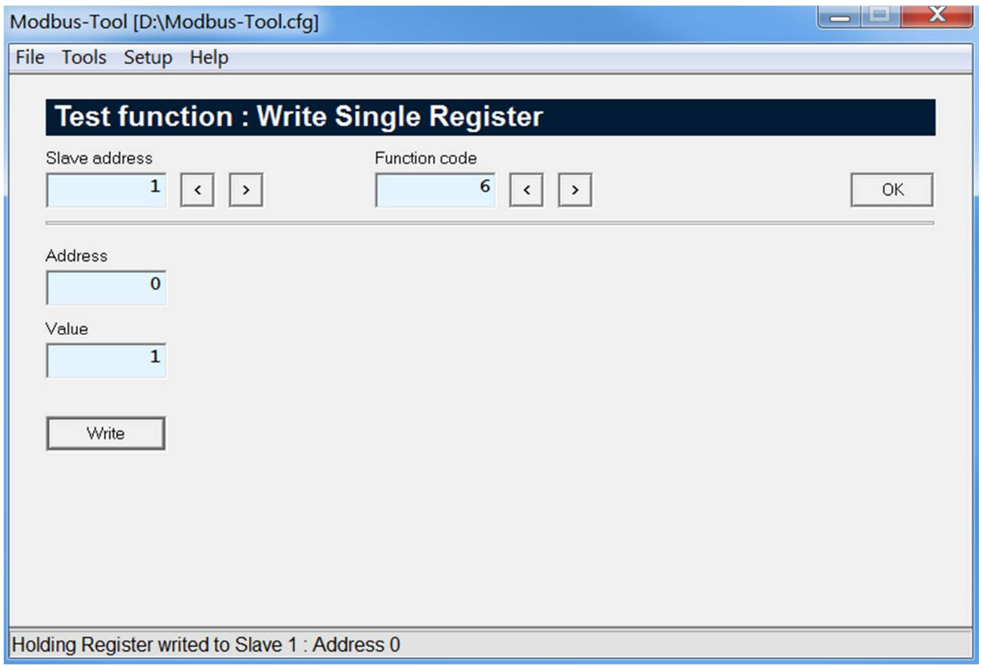
Quantity: 1

Address > Bit

```
00000 > 0
```

Coil(s) readed from Slave 1 : Address 0 : Quantity 1

For the writing functions, a similar procedure must be used. After setting the values in the boxes, use the **"Write"** command to activate the writing on the slave module via the protocol command. For example, for writing a single register, the screen layout is as follows:



Test function : Write Single Register

Slave address: 1

Function code: 6

Address: 0

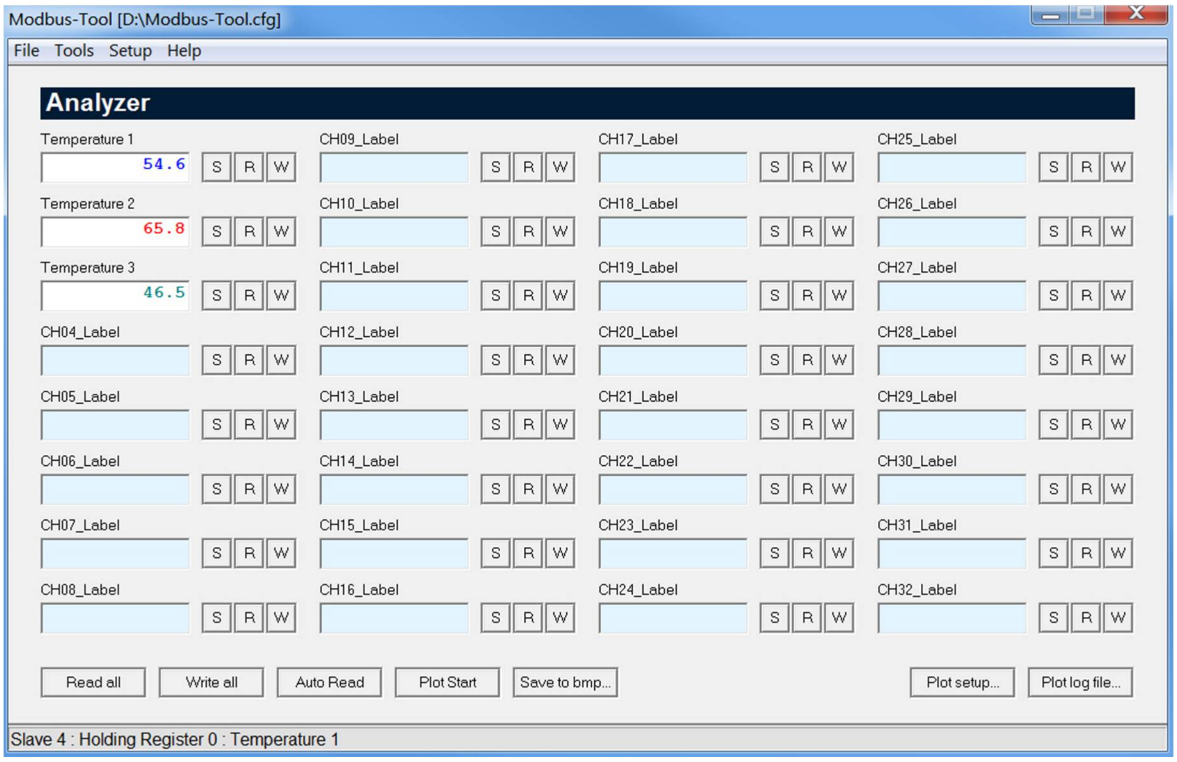
Value: 1

Holding Register writed to Slave 1 : Address 0

NOTE: Not all of the Modbus function codes are implemented in the tool. For all commands that are not available the text "Test function: not implemented" is displayed.

2.5. “Analyzer” menu

This tool is a flexible 32 channels acquisition system that can be used to view and edit Modbus register values with "Input register" or "Holding register" type. Each channel can be associated to a specific register and slave address and so the tool allows operations on a set of different devices connected to the network.
Each channel has a value box and three buttons:


Modbus-Tool [D:\Modbus-Tool.cfg]

Analyzer

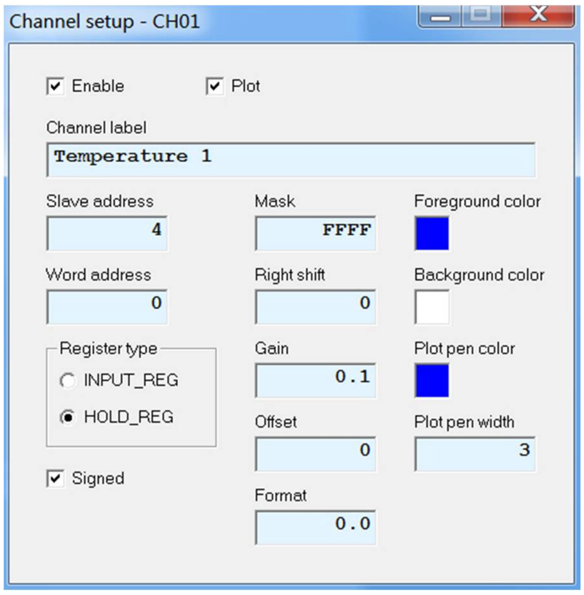
Temperature 1: 54.6	CH09_Label	CH17_Label	CH25_Label
Temperature 2: 65.8	CH10_Label	CH18_Label	CH26_Label
Temperature 3: 46.5	CH11_Label	CH19_Label	CH27_Label
CH04_Label	CH12_Label	CH20_Label	CH28_Label
CH05_Label	CH13_Label	CH21_Label	CH29_Label
CH06_Label	CH14_Label	CH22_Label	CH30_Label
CH07_Label	CH15_Label	CH23_Label	CH31_Label
CH08_Label	CH16_Label	CH24_Label	CH32_Label

Slave 4 : Holding Register 0 : Temperature 1

The **"S"** button activates the channel configuration window, the **"R"** button allows a single reading of the value while the **"W"** button writes the value previously typed in the box. In this tool some buttons are available for contemporary reading (**"Read all"**) and writing (**"Write all"**) of all enabled channels. The **"Auto Read"** button enables continuous reading of the values according to the sampling time set by **"Sampling time"** parameter under the "Setup" menu.

2.5.1. Channel configuration

Before using the tool, the desired channels must be configured. This is accomplished using the **"S"** button, next to each box for the value visualization, that activates the channel setup window:



[✓] Enable [✓] Plot

Channel label: Temperature 1

Slave address: 4

Word address: 0

Register type: ( ) INPUT_REG [✓ HOLD_REG]

[✓] Signed

Mask: FFFF

Right shift: 0

Gain: 0.1

Offset: 0

Format: 0.0

Foreground color: 🟦

Background color: ⬜

Plot pen color: 🟦

Plot pen width: 3

It's very important the enabling (by the **"Enable"** option) of only channels needed because, by doing so, the update time is significantly reduced. A channel can also be included in the graph (Trend) enabling the **"Plot"** option. The **"Channel label"** field is used to describe the channel with a string that will be displayed above the channel box and in the plotting legend.

The channel must be associated with one specific Modbus register ("Input register" or "Holding register") of a slave module with a given network address. Three fields are available for this pourpose: **"Slave address"**, **"Word address"** and **"Register type"**. The **"Signed"** option allows to consider the value as signed word for which the values 0÷65535 are automatically translated in the range -32768÷32767 in reading and vice versa in writing operations.

The **"Mask"**, **"Right shift"**, **"Gain"**, **"Offset"** and **"Format"** parameters allow to appropriately process the reading of the word, before it is displayed in the box, using the following procedure:

Visualized value = ((Word AND Mask) >> Shift) * Gain + Offset

Some examples are shown in the following tables:

Word	Mask = FFFF	Right shift = 0	Gain = 0.01	Offset = 10	Format = 0.00
320	320	320	3.2	13.2	13.20

Word	Mask = FF00	Right shift = 8	Gain = 0.33	Offset = 10.5	Format = 0.0
1025	1024	4	1.32	11.82	11.8

Word	Mask = 0100	Right shift = 8	Gain = 1	Offset = 0	Format = 0
258	256	1	1	1	1

In the last example, the 0/1 status of bit 8 of the word is visualized and so the channel is used for displaying a Boolean value inserted in a word register.
The procedure applied to the register word value reading, is used, inversely, in the writing of a new value into the register:

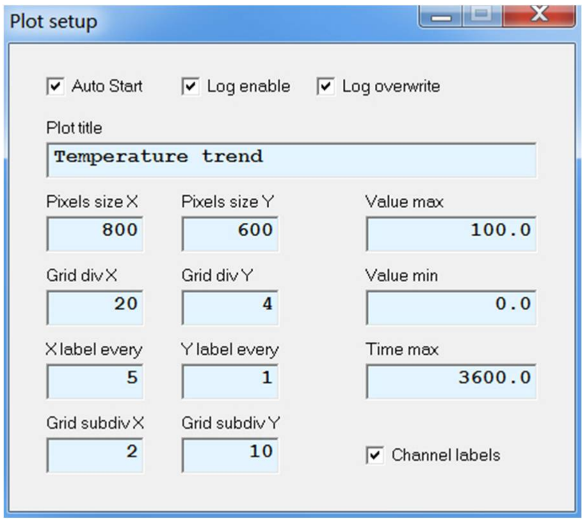
Word = (((Set value – Offest) / Gain) << Shift) AND Mask

However it must be considered that the reverse process with the AND Mask operation does not take account of the value of the bits outside the mask (set to zero) for which the entire value of the word, to be written, cannot be reconstructed except in the inner part of the mask. Furthermore, the operation carried out by Mask and Shift has meaning only with the "Signed" option disabled.

With the channel setup it's also possible to customize the **colors** of the text of the value, the viewing box and the description label reported in the chart legend. For the graph the thickness in pixels of the pen can also be set using the **"Plot pen width"** parameter.

2.5.2. Trend chart and logging file creation

The **"Auto Read"** button enables continuous reading of values in channel boxes. The automatic reading ends by again clicking the "Auto Read" button. The channel values, while capturing in this way, can also be reported, in real time, on a chart to show trends over time. For this function, the plotting window must be configured using **"Plot Setup"** button:


Plot setup

☑ Auto Start ☑ Log enable ☑ Log overwrite

Plot title: Temperature trend

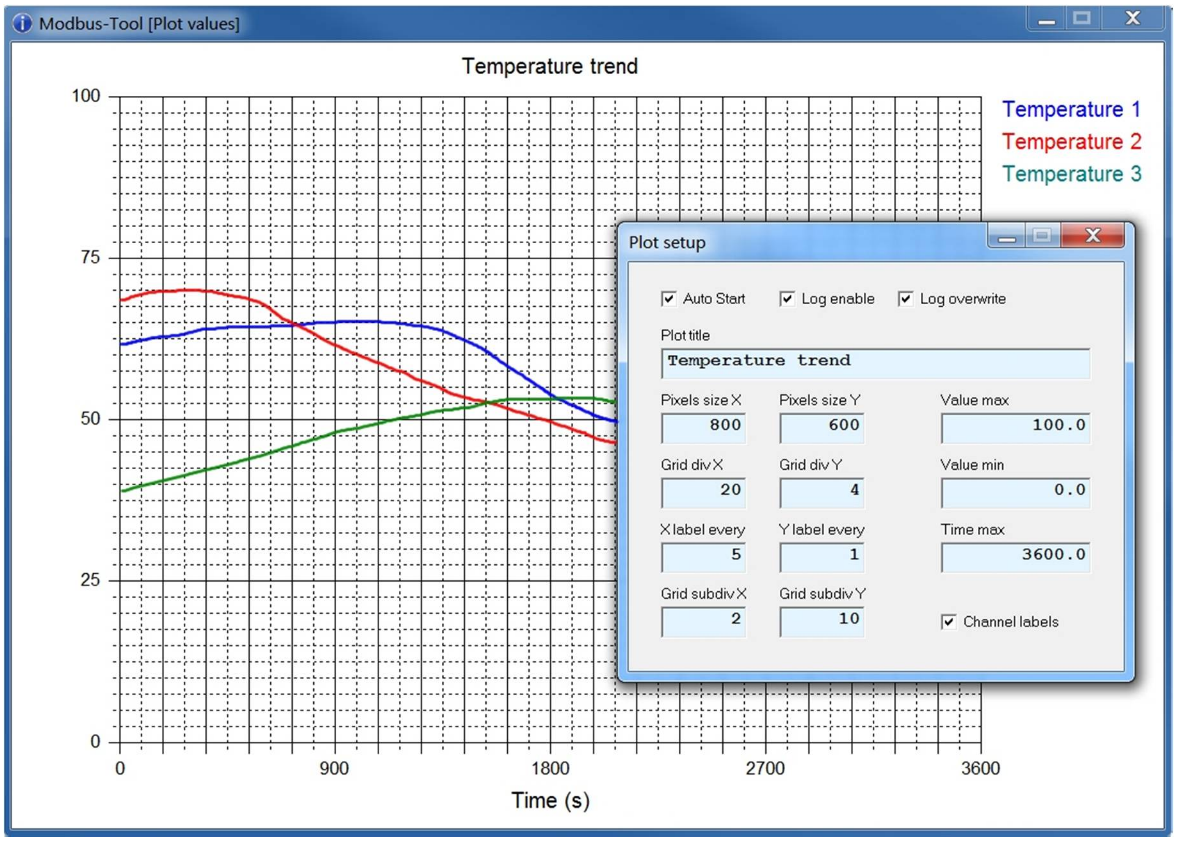
Pixels size X: 800	Pixels size Y: 600	Value max: 100.0
Grid div X: 20	Grid div Y: 4	Value min: 0.0
X label every: 5	Y label every: 1	Time max: 3600.0
Grid subdiv X: 2	Grid subdiv Y: 10	☑ Channel labels

The "Auto Start" option allows the automatic activation of the plotting whenever the continuous acquisition begins with "Auto Read" button. Without this option, the chart can still be switched on/off manually with the **"Plot Start/Stop"** button several times at various periods of continuous reading. Once created, the chart can be saved as a bitmap image on a selected file using the **"Save to bmp"** button, available directly on the main page of the tool.

The values acquired and displayed on the graph can also be saved to a file in CSV format, to be subsequently analyzed for example with Excel. This requires the **"Log enable"** option and the selection of a file name using the "Plot log files" button, available on the main page of the tool. The **"Log overwrite"** option allows the saving of the values always on the same selected file, otherwise it will automatically generate the name of a new file for each chart created.

The **"Plot title"** field of the setup window allows the definition of the chart title. The **"Pixel size X/Y"** values define the size of the graphic window (in pixels) to create graphs with different resolutions also usefull for the bitmap files saving. The Y axis range is defined by **"Value min/max"** fields while the X axis range by the **"Time Max"** parameter. Other fields are used to select the **main and secondary grid** of the two axes and the presence of the axes labels.

The result of the Plot window, using the values of configuration example, is the following:



☑ Auto Start ☑ Log enable ☑ Log overwrite

Plot title: Temperature trend

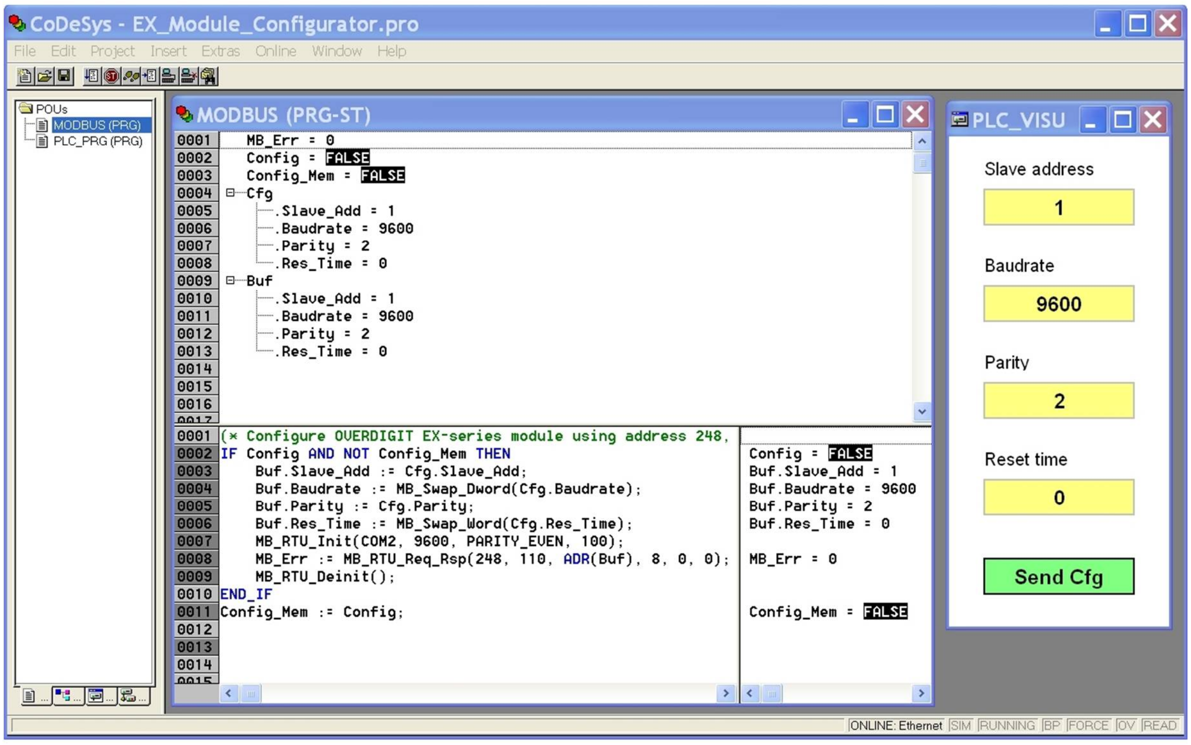
Pixels size X: 800	Pixels size Y: 600	Value max: 100.0
Grid div X: 20	Grid div Y: 4	Value min: 0.0
X label every: 5	Y label every: 1	Time max: 3600.0
Grid subdiv X: 2	Grid subdiv Y: 10	☑ Channel labels

3. Configuring with CoDeSys application

For the configuration of EX series modules some applications running directly on the PLC can also be used.
The following section shows a CoDeSys example program of how to set all the communication parameters. In addition on the **MODBUS_EX.lib** library, concerning the modules of the series, there is a function block for the immediate setting of the slave address of the modules, using, for the other parameters, the informations extracted from the "PLC Configuration".

3.1. "EX_Module_Configurator" program

This simple program is an utility service, to run on the PLC, for configuration of communication parameters for EX series modules. For this purpose it's used the **MB_RTU_Req_Rsp** master function of **MODBUS_Lib.lib** library, specific for *OVERDIGIT* PLCs.
This is the basic function of master communication used by all specific functions of the various protocol command codes.


CoDeSys - EX_Module_Configurator.pro
File Edit Project Insert Extras Online Window Help
POUs
MODBUS (PRG)
PLC_PRG (PRG)
MODBUS (PRG-ST)
PLC_VISU
ONLINE: Ethernet SIM RUNNING BP FORCE OV READ

```
MB_Err = 0
Config = FALSE
Config_Mem = FALSE
⊟ Cfg
    .Slave_Add = 1
    .Baudrate = 9600
    .Parity = 2
    .Res_Time = 0
⊟ Buf
    .Slave_Add = 1
    .Baudrate = 9600
    .Parity = 2
    .Res_Time = 0
```

```
(* Configure OVERDIGIT EX-series module using address 248,
IF Config AND NOT Config_Mem THEN
    Buf.Slave_Add := Cfg.Slave_Add;
    Buf.Baudrate := MB_Swap_Dword(Cfg.Baudrate);
    Buf.Parity := Cfg.Parity;
    Buf.Res_Time := MB_Swap_Word(Cfg.Res_Time);
    MB_RTU_Init(COM2, 9600, PARITY_EVEN, 100);
    MB_Err := MB_RTU_Req_Rsp(248, 110, ADR(Buf), 8, 0, 0);
    MB_RTU_Deinit();
END_IF
Config_Mem := Config;
```

```
Config = FALSE
Buf.Slave_Add = 1
Buf.Baudrate = 9600
Buf.Parity = 2
Buf.Res_Time = 0

MB_Err = 0


Config_Mem = FALSE
```

Slave address	1
Baudrate	9600
Parity	2
Reset time	0

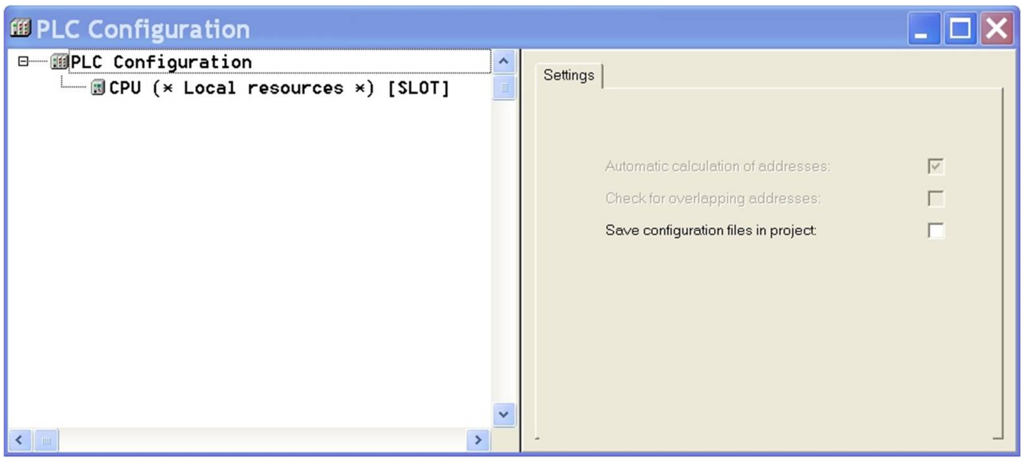
A visualization form operates as interface for the introduction of the configuration values to be sent to the slave module. After activation, with the PG button of module, of the known and provisory (address 248, baudrate 9600, even parity) communication parameters, clicking "Send Cfg" button, the instructions included in the IF statement are executed for a single time.
The entered values are packed in 8 bytes array that acts as a data area buffer for the function, with command code 110, for sending/receiving frames. At the end the communication is de-initialized.

3.2. Configuration and using via “MODBUS_EX.lib” library

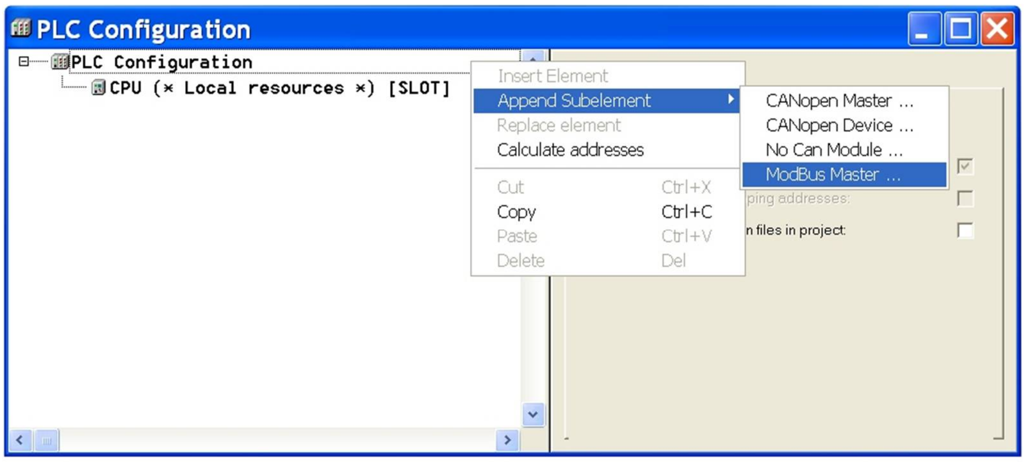
With the EX series modules it’s supplied a specific CoDeSys library containing all needing for immediate integration of the modules in the application via the **"PLC Configuration"** of development environment.

The "PLC Configuration" of CoDeSys allows the composition of the whole I/O system of PLC by inserting the various expansion modules directly from a menu list. In order for the development environment to know the particular characteristics of the module (for example, how many Inputs/Outputs and what type) it’s supplied a configuration file, with the **.CFG** extension, specific for each module. The configuration files of the EX series modules are supplied pre-installed in the targets of *OVERDIGIT* PLCs.

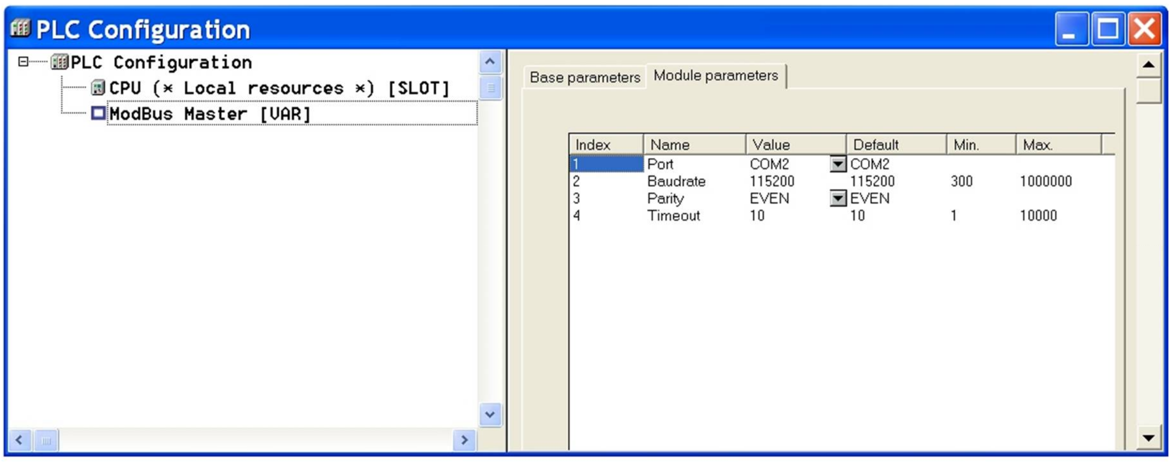
Opening the "PLC Configuration" of a new project, the basic I/O structure is visualized still without any module inserted:


PLC Configuration
PLC Configuration
CPU (* Local resources *) [SLOT]
Settings
Automatic calculation of addresses: ✓
Check for overlapping addresses:
Save configuration files in project:

After selecting the root item of the tree ("PLC Configuration" line), with the right mouse button, the context menu can be called for inserting, as Subelement, of the master module of communication indicated in the list with **"Modbus Master"**:

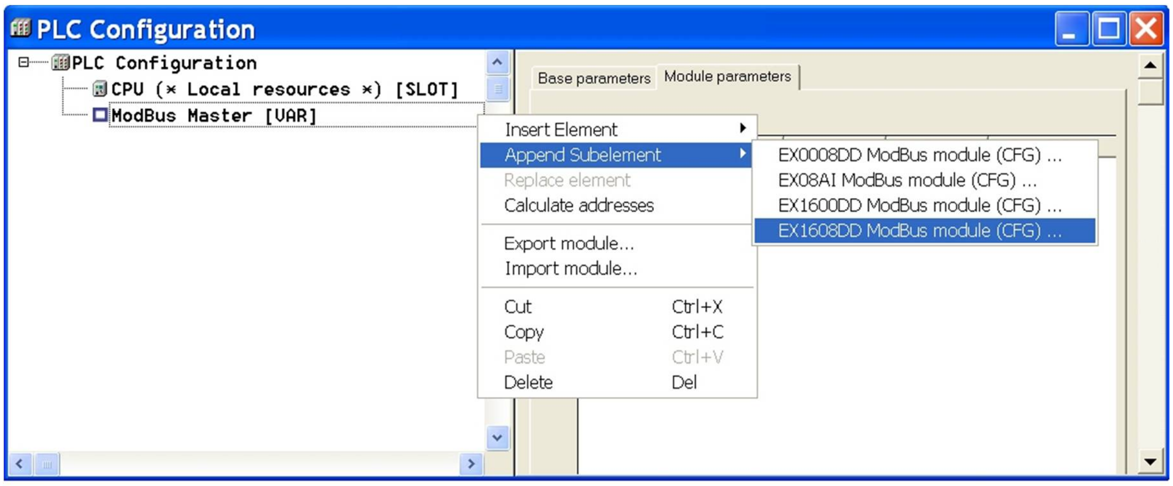


The master module allows the setting, using the **"Module parameters"** Tab, of the Modbus communication configuration:

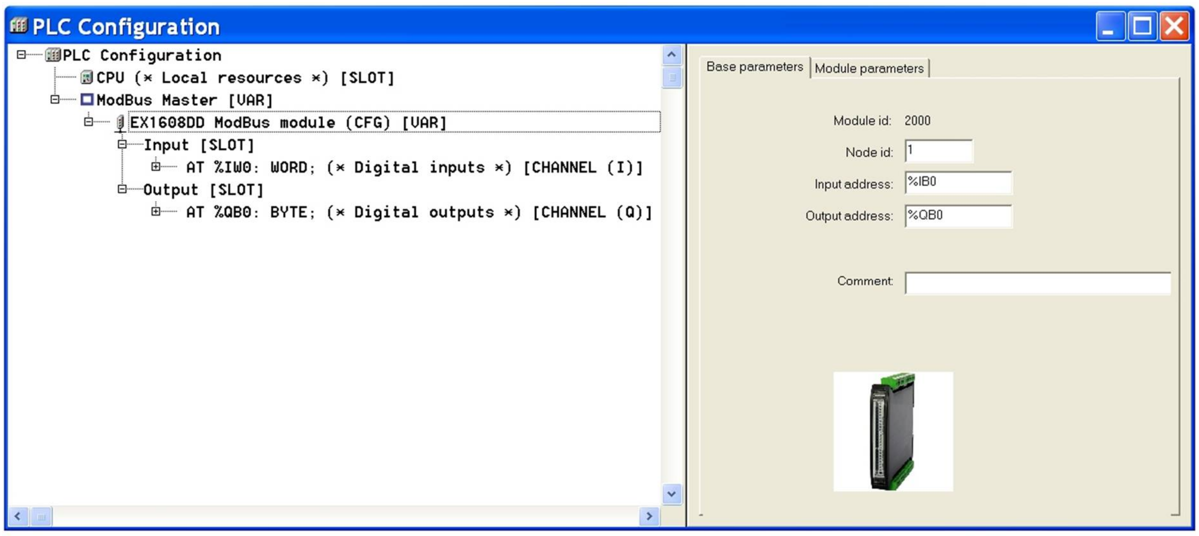


Index	Name	Value	Default	Min.	Max.
1	Port	COM2	COM2		
2	Baudrate	115200	115200	300	1000000
3	Parity	EVEN	EVEN		
4	Timeout	10	10	1	10000

The **"Port"** field indicates the serial port used by the specific master PLC to manage the RS485 network, while **"Baudrate"** and **"Parity"** parameters are used to set the master communication and to be transferred to the slaves during their configuration.
Finally, the **"Timeout"** parameter is used by the master to set, for the communication functions, the maximum time to wait for a response from the slave.
At this point, all slave modules that will be used on the expansion network must be entered as Subelement of "Modbus Master" module:



In this example only one EX1608DD slave module is loaded. This module has 16 inputs and 8 outputs (digital type), whose resources are associated with the IW0% word of Input area and the %QB0 byte of Output area, as shown in the structure tree:



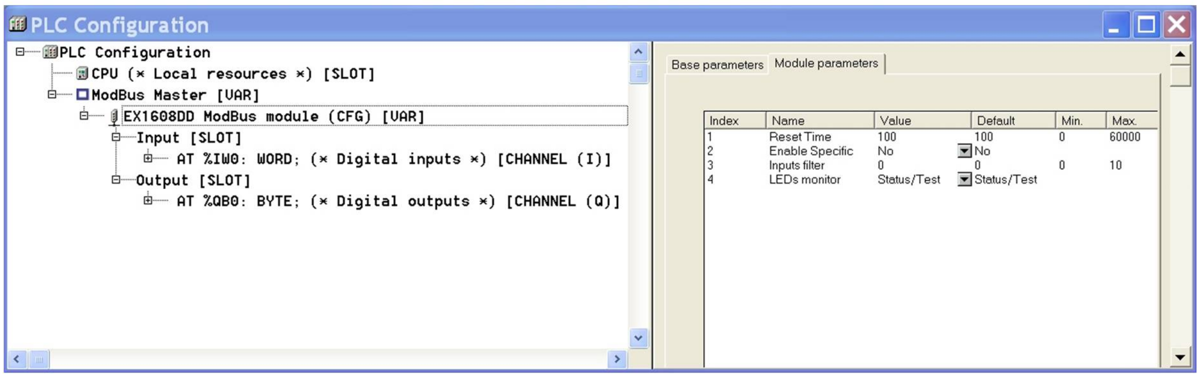
```
PLC Configuration
⊟─PLC Configuration
   ├─CPU (* Local resources *) [SLOT]
   ⊟─ModBus Master [VAR]
      ⊟─ EX1608DD ModBus module (CFG) [VAR]
         ⊟─Input [SLOT]
         │  ⊞─ AT %IW0: WORD; (* Digital inputs *) [CHANNEL (I)]
         ⊟─Output [SLOT]
            ⊞─ AT %QB0: BYTE; (* Digital outputs *) [CHANNEL (Q)]
```

Base parameters | Module parameters

Module id: 2000
Node id: 1
Input address: %IB0
Output address: %QB0
Comment:

NOTE: Placing more slave modules, the system will automatically assign the resources to the Inputs and Outputs areas at successive addresses, but in different ways depending on the setting of the "Byte addressing mode" in the "Target Settings>General" menu.

For each module, the address of the slave must be manually inserted. For this the **"Node id"** field must be edited after selecting the specific slave module in the tree. It's recommended, but not mandatory, to number the slaves progressively starting from 1.
Opening the "Module parameters" Tab of the slave module, any its specific parameter is displayed:



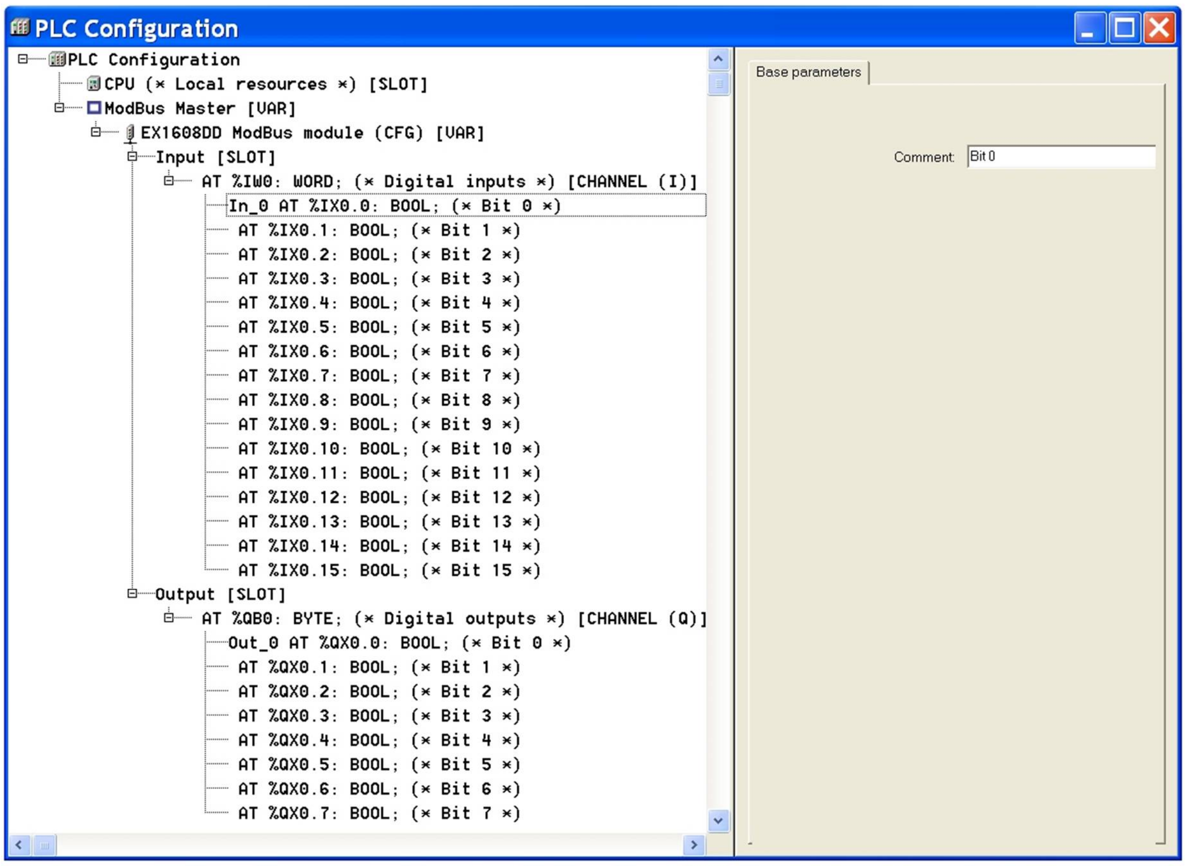
```
PLC Configuration
⊟─PLC Configuration
   ├─CPU (* Local resources *) [SLOT]
   ⊟─ModBus Master [VAR]
      ⊟─ EX1608DD ModBus module (CFG) [VAR]
         ⊟─Input [SLOT]
         │  ⊞─ AT %IW0: WORD; (* Digital inputs *) [CHANNEL (I)]
         ⊟─Output [SLOT]
            ⊞─ AT %QB0: BYTE; (* Digital outputs *) [CHANNEL (Q)]
```

Base parameters | Module parameters

Index	Name	Value	Default	Min.	Max.
1	Reset Time	100	100	0	60000
2	Enable Specific	No	No		
3	Inputs filter	0	0	0	10
4	LEDs monitor	Status/Test	Status/Test		

In the example of the EX1608DD module the **"Reset Time"** parameter is available and this is normally present on all devices with outputs. This parameter configures the safety timer for the automatic deactivation of the outputs in the absence of proper communication on the bus.
Since the **V23.9.46.1 release** of the IEC-line software package, the "PLC Configuration" also provides the setting of additional parameters specific for the slave module. Refer to the modules documentation for more details on the "Holding Registers" used for the configuration. The **"Enable Specific"** parameter sets a flag to inform the MB_RTU_Slave_Cfg function block, illustrated in the following, that also all parameters, listed after the same, must be transferred to the module.

Finally, the "PLC Configuration" allows the defining of symbolic names for all individual I/O resources, mapped to absolute addresses in the Input and Output areas. For this purpose the screen area at the left of the AT text of the resource must be clicked:



```
PLC Configuration
    CPU (* Local resources *) [SLOT]
    ModBus Master [VAR]
        EX1608DD ModBus module (CFG) [VAR]
            Input [SLOT]
                AT %IW0: WORD; (* Digital inputs *) [CHANNEL (I)]
                    In_0 AT %IX0.0: BOOL; (* Bit 0 *)
                    AT %IX0.1: BOOL; (* Bit 1 *)
                    AT %IX0.2: BOOL; (* Bit 2 *)
                    AT %IX0.3: BOOL; (* Bit 3 *)
                    AT %IX0.4: BOOL; (* Bit 4 *)
                    AT %IX0.5: BOOL; (* Bit 5 *)
                    AT %IX0.6: BOOL; (* Bit 6 *)
                    AT %IX0.7: BOOL; (* Bit 7 *)
                    AT %IX0.8: BOOL; (* Bit 8 *)
                    AT %IX0.9: BOOL; (* Bit 9 *)
                    AT %IX0.10: BOOL; (* Bit 10 *)
                    AT %IX0.11: BOOL; (* Bit 11 *)
                    AT %IX0.12: BOOL; (* Bit 12 *)
                    AT %IX0.13: BOOL; (* Bit 13 *)
                    AT %IX0.14: BOOL; (* Bit 14 *)
                    AT %IX0.15: BOOL; (* Bit 15 *)
            Output [SLOT]
                AT %QB0: BYTE; (* Digital outputs *) [CHANNEL (Q)]
                    Out_0 AT %QX0.0: BOOL; (* Bit 0 *)
                    AT %QX0.1: BOOL; (* Bit 1 *)
                    AT %QX0.2: BOOL; (* Bit 2 *)
                    AT %QX0.3: BOOL; (* Bit 3 *)
                    AT %QX0.4: BOOL; (* Bit 4 *)
                    AT %QX0.5: BOOL; (* Bit 5 *)
                    AT %QX0.6: BOOL; (* Bit 6 *)
                    AT %QX0.7: BOOL; (* Bit 7 *)
```

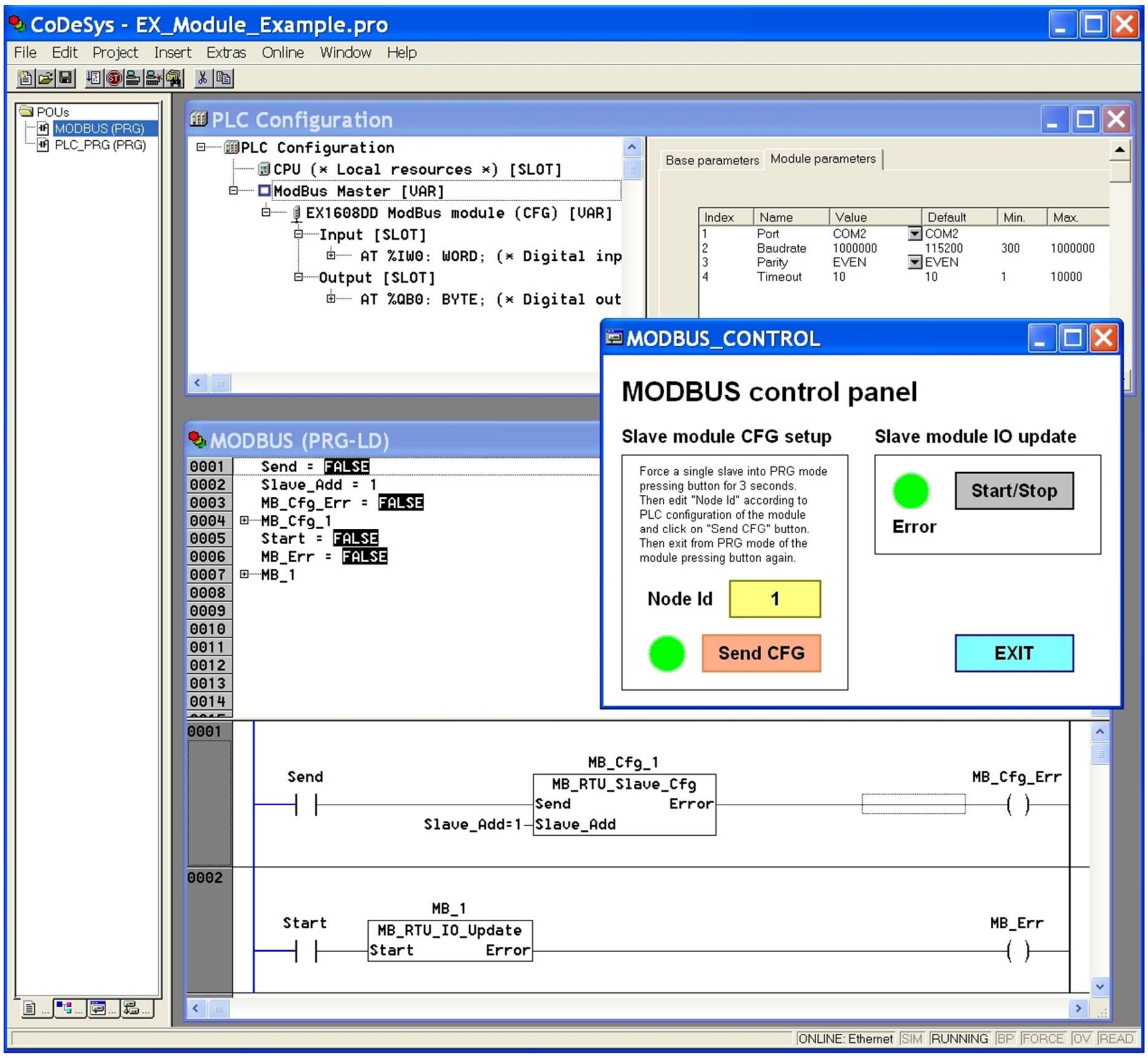
Comment: Bit 0

At this point the "PLC Configuration" has all the information on the specific structure of the I/O resources and how they should be updated on the fieldbus using the Modbus protocol. The information on the "PLC Configuration" become an integral part of the specific application and are saved in the program. This information, however, do not generate a specific executable code but are available to parameterize the IEC program.
The functions of **"SysLibPLCConfig.lib"** CoDeSys library allow the extraction of the information available in the "PLC Configuration". On these functions is based **"MODBUS_EX.lib"** library for EX series modules support. This library contains two function blocks.

The **MB_RTU_Slave_Cfg** function block allows to sending all values of communication configuration to a slave module, with the programming mode activated by the PG button. Regarding the Baudrate and Parity, the values, inserted in the "PLC Configuration" under the parameterization of the master module, are applied as it is obvious that all the slaves should use the same parameters of the master. The value of any specific parameter of the module (for example the "Reset Time") is instead derived from the parameterization of individual slave modules inserted.
If the “Enable Specific” parameter is set to Yes, also all parameters, subsequent to this, are transferred to the module by the function block.

Finally, the configuration function block accepts as input the slave address (Node id) to be provided directly in accordance with the settings, in "Basic parameters" Tab, of the slave module. The "Send" input bit activates the transmission of configuration command.

The **MB_RTU_IO_Update** function block is the "engine" of the Modbus master communication to update the I/O of all slave modules inserted. In this case, all the necessary information is derived automatically from the settings used in the "PLC Configuration":


CoDeSys - EX_Module_Example.pro
File Edit Project Insert Extras Online Window Help
POUs
MODBUS (PRG)
PLC_PRG (PRG)
PLC Configuration
Base parameters
Module parameters
MODBUS_CONTROL
MODBUS (PRG-LD)
ONLINE: Ethernet SIM RUNNING BP FORCE OV READ

```
PLC Configuration
    CPU (* Local resources *) [SLOT]
    ModBus Master [VAR]
        EX1608DD ModBus module (CFG) [VAR]
            Input [SLOT]
                AT %IW0: WORD; (* Digital inp
            Output [SLOT]
                AT %QB0: BYTE; (* Digital out
```

Index	Name	Value	Default	Min.	Max.
1	Port	COM2	COM2		
2	Baudrate	1000000	115200	300	1000000
3	Parity	EVEN	EVEN		
4	Timeout	10	10	1	10000

MODBUS control panel

Slave module CFG setup

Force a single slave into PRG mode pressing button for 3 seconds. Then edit "Node Id" according to PLC configuration of the module and click on "Send CFG" button. Then exit from PRG mode of the module pressing button again.

Node Id: 1

Send CFG

Slave module IO update

Start/Stop

Error

EXIT

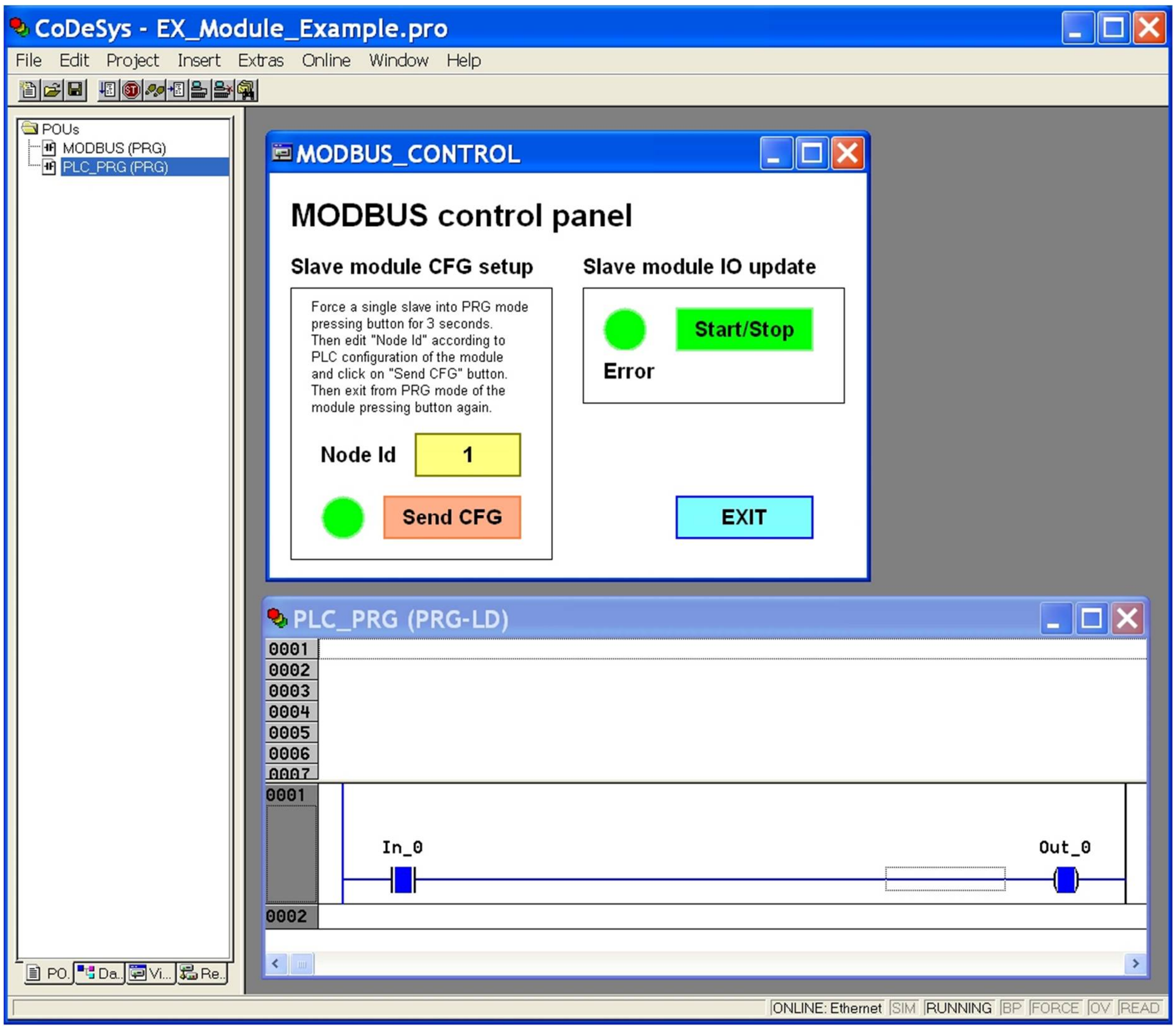
```
Send = FALSE
Slave_Add = 1
MB_Cfg_Err = FALSE
MB_Cfg_1
Start = FALSE
MB_Err = FALSE
MB_1
```

```
0001
        Send                       MB_Cfg_1                                MB_Cfg_Err
                               MB_RTU_Slave_Cfg
    ---| |---------------------|Send         Error|----------------------( )---
                   Slave_Add=1-|Slave_Add         |

0002
                         MB_1
        Start       MB_RTU_IO_Update                                       MB_Err
    ---| |---------|Start       Error|------------------------------------( )---
```

The MB_RTU_IO_Update function block requires the "Start" input bit to enable communication and it must be inserted in a POU executed periodically, according to the repetition time chosen for the sampling of modules.
For I/O update the block calls the basic function of the Modbus communication on MODBUS_Lib.lib library because it's used the **custom function code 102**, a command implemented in the EX series modules. With this single command both Input and Outputs areas of the slave are updated in one frames exchange. All output resources of module are inserted into an array of bytes and sent via

the data area of the command, while the data area, received from the slave, will return an array of bytes containing all the input resources.
The MODBUS_EX.lib library is supplied as "open source" and is also a base example for custom versions.
The end result of the IEC program in execution is as follows:



Force a single slave into PRG mode pressing button for 3 seconds. Then edit "Node Id" according to PLC configuration of the module and click on "Send CFG" button. Then exit from PRG mode of the module pressing button again.

The visualization form allows the communication configuration for the slave modules and the enabling of the continuous updating of the I/O. A simple ladder diagram POU uses an input bit of expansion module and copies it on an output bit of it.