Energy XT PRO
Communication *Protocols*

XT Energy PRO

CONTENTS

1 USE OF MANUAL

In order to refer to the manual quickly and easily, customers may find the following useful:

Call-outs

Callout column:
Callouts on the topics described are placed to the left of the text to allow the user to find the required information quickly.

Cross references

Cross references:
All the words in *italics* are listed in the index with a reference to the page where they are described in more detail;
the text below serves as an example:
"activation of the alarm stops the *compressors*"
The italics indicate that under Compressors in the index there is a reference to the page where compressors are described in more detail.
If the online Help on the PC is used, the words in italics become proper hyperlinks (automatic links activated by a click of the mouse) that connect the different sections in the manual and allow you to navigate through the document.

Highlighted icons:

Some parts of the text are highlighted in the callout column using icons with the following meanings:

Note: draws attention to a specific topic that users should take into account.

Tip: highlights a suggestion that helps users to understand and *use* the information on the topic described.

Attention! : **highlights**

1. **information that may damage the system or place persons, equipment, data, etc at risk if not known. These sections must always be read prior to *use*.**
2. **a specific topic that users should take into account so that the system does not malfunction or is used improperly.**

2 UART SERIALS ON ENERGY XT-PRO

UART serials

The Energy XTPRO has two UART communication ports named *COM1* and *COM3*.

COM1

***COM1*:**

It is a RS485 serial with signals RS485+,RS485- and RS485GND

COM3

***COM3*:**
It is a RS232 serial with signals TX, RX, CTS, RTS and DTR (fixed). It can be accessed, although in different ways, via a DB9 connector and a MOLEX connector (situated next to COM4) that "returns" the signals TX,RX and RTS **to TTL electrical level**.

"param." table COM1 and COM3:

The *COM1* and *COM3* "parameterization" list is below:

Parameter Name	*Modbus* Address [DEC]	Description
PAR_ANA_BIOS_187	218	Device serial address family
PAR_ANA_BIOS_190	222	*COM1* protocol selection 2=*Micronet* 3=*Modbus*/RTU
PAR_ANA_BIOS_191	223	*COM1* baud selection 0=9600 b/s 1=19200 b/s 2=38400 b/s
PAR_ANA_BIOS_192	224	*COM1* parity selection 0=null 1=odd 2=even
PAR_ANA_BIOS_193	225	*COM3* protocol selection 0=*Televis* 1=*Televis* MODEM 2=*Micronet* 3=*Modbus*/RTU 4=*Modbus*/ASCII
PAR_ANA_BIOS_194	226	*COM3* baud selection 0=9600 b/s 1=19200 b/s 2=38400 b/s
PAR_ANA_BIOS_195	227	*COM3* parity selection 0=null 1=odd 2=even
PAR_BOO_BIOS_19	228	Selection 7/8 data bits *COM3* 0=7 data bits 1=8 data bits
PAR_BOO_BIOS_20	229	RTS signal management 0=normal management: -12V received,+12V transmitted 1=always +12V to supply external converters RS232-RS485

The board address is the same for serial *COM1* and serial *COM3*.
It is a byte consisting of 2 parts:

- The MSB nibble is the device family and is a parameter saved in the EEPROM (PAR_ANA_BIOS_187);
- The LSB nibble is the device address that is read using the three dip switches DIP2,3,4

- For example if J2=ON,J3=OFF,J4=OFF the LSB nibble will be 1
- For example if J2=ON,J3=OFF,J4=OFF the LSB nibble will be 3

NB: See the *Hardware Address* sub-section in the attached "*Troubleshooting*" section of the Appendix.

The operating protocol on serials *COM1* and *COM3* and the serial address of the device depends on the operating mode of the Energy XTPRO. The protocol changes on *COM1* after the first 15 seconds after start-up without problems of the XTPRO.

N.B.: To *use* port RS 485 in *WORKBENCH* mode while downloading TIC, the serial port RS 232 (and also TTL) must not be active. Otherwise the TIC download may not complete successfully.

Hence, while downloading the TIC, make sure the MASTER *MODBUS* mb03req() function with AutoScan=TRUE hasn't been used (or is still active) as BIOS would retrieve it and run it in the background even after *WORKBENCH* is stopped. Also check that an external supervisor is not querying Energy XT-PRO with *MODBUS* protocol on serial port *COM3*.

Users are responsible for preventing this from occurring before downloading the application. Otherwise, simply *use* the recovery card before starting the download.

The next two sections explain what has been stated here.

Start-up without card

2.1 COM1 and COM3 vs start-up without IIC card

When it is switched on the instrument checks for any HW problems and makes sure that all the data necessary for launching the application is present.
If the data in the external FLASH is not available or when the FLASH, external RAM or EEPROM are not usable, the following error messages are displayed:

- product codes and external FLASH device code incorrect ERR[1].
- External RAM check error ERR[2].
- Linker table programming error ERR[3].
- Menu descriptor programming error ERR[4].
- Corrupt unrecoverable external EEPROM ERR[5].
- TIC programming error ERR[6].

If none of these anomalies occurs, normal start-up occurs.

These are the settings for *COM1* and *COM3* in these situations:

Situation	Error	Mode	RS232			RS485		
			Protocol	Address	Par.	Protocol	Address	Par.
Application does not exist or is incomplete	ERR[3], ERR[4] or ERR[6]	Energy XT-Pro "in Emergency" mode	*Televis*	Dip-Switch	19200, E, 8, 1	ISaGRAF	Dip-Switch	19200, N, 8, 1
HW Problems	ERR[1], ERR[2] or ERR[5]		*Televis*	Dip-Switch	19200, E, 8, 1	UNet	Dip-Switch	9600,x, 8, 1
Normal start-up	None	Energy XT-Pro "Pre-programmed"	From Parameter	Dip-Switch+ FAMILY	From parameter	ISaGRAF in first 15 secs. after start-up.	Dip-Switch	19200, N, 8, 1
						Selection from parameter if ISaGRAF connection is not established within 15 sec.	Dip-Switch+ FAMILY	From Parameter

- **ERR[1], ERR[2] or ERR[5] presence enable always Unet protocol (Diop-switch 9600, x, 8, 1**

NOTE: in the table ISaGRAF coincides with the *Workbench*

If the Energy XTPRO is communicating with the parameter-selected protocol on *COM1*, you can go to the *Workbench* communication protocol (Debug Mode ON) and vice versa (Debug Mode OFF) using the keyboard (EXTK PRO or EXTU PRO), going to the SERVICE menu and selecting the correct function.

2.2 COM1 and COM3 vs start-up with IIC card

start-up without IIC recovery card

If the instrument identifies the RECOVERY CARD at start-up *COM1* and *COM3* do as follows:

IIC CARD	Situation	Error	Mode	RS232			RS485		
				Protocol	Address	Par.	Protocol	Address	Par.
Security Code	HW Problems	ERR[3], ERR[4] or ERR[6] + If HW problems ERR[1], ERR[2] or ERR[5]	Energy XT-Pro "in Emergency" mode	*Televis*	Dip-Switch	19200, E, 8, 1	UNet	Dip-Switch	9600, x, 8, 1
	NO HW problems						ISaGRAF	Dip-Switch	19200,N,8,1

NOTE: in the table ISaGRAF coincides with the *Workbench* and the security code indicates the RECOVERY card.

2.3 "COM1" serial (RS485)

2.3.1 Use

This is used to connect the Energy XTPRO to the outside world but it is also the only place where the *WORKBENCH* can be connected to the Energy XT and communicate with it. This serial can be used to perform operations on the internal and external flash of the microcontroller for downloading the BIOS-*WORKBENCH* interface tables, navigation menu and *WORKBENCH* application programme (TIC+BD). It is a "slave" serial and therefore packages are not spontaneously emitted from this serial but only response packages according to the *protocols* indicated below.

The communication speed and parity will be controlled by the HW and the protocol used as far as possible.

2.3.2 Protocols Usable on "COM1"

WORKBENCH

WORKBENCH

For connection of the instrument and the *WORKBENCH*. Also used for downloading the BIOS-*WORKBENCH* interface tables, navigation menu and *WORKBENCH* application programme (TIC+BD).

Micronet

Micronet

For connecting the instrument as SLAVE to an RS485 network containing devices such as TelevisCompact, *Televis* (via PC Interface), or ParamManager as MASTER host.

MODBUS

MODBUS

For connecting the instrument as SLAVE to an RS485 network containing any *MODBUS* HOST as MASTER host (also on PC). The *MODBUS* protocol will only be an RTU type with a fixed baud rate of 9600 bps

2.3.3 COM1 PARAMETERIZATION FOR WORKBENCH PROTOCOL

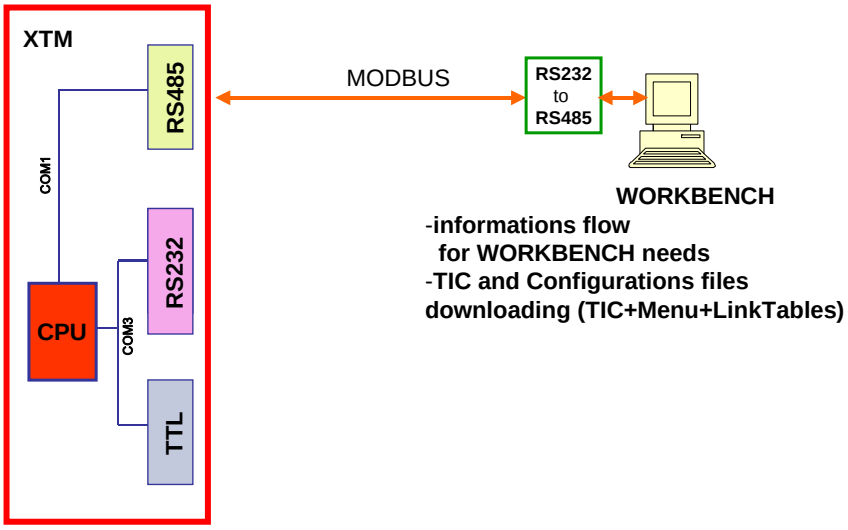
*COM1*_BAUD	19200
*COM1*_PARITY	null
*COM1*_DATA	8
*COM1*_STOP	1

2.3.4 COM1 PARAMETERIZATION FOR Micronet and MODBUS PROTOCOLS (parameters in EEPROM highlighted)

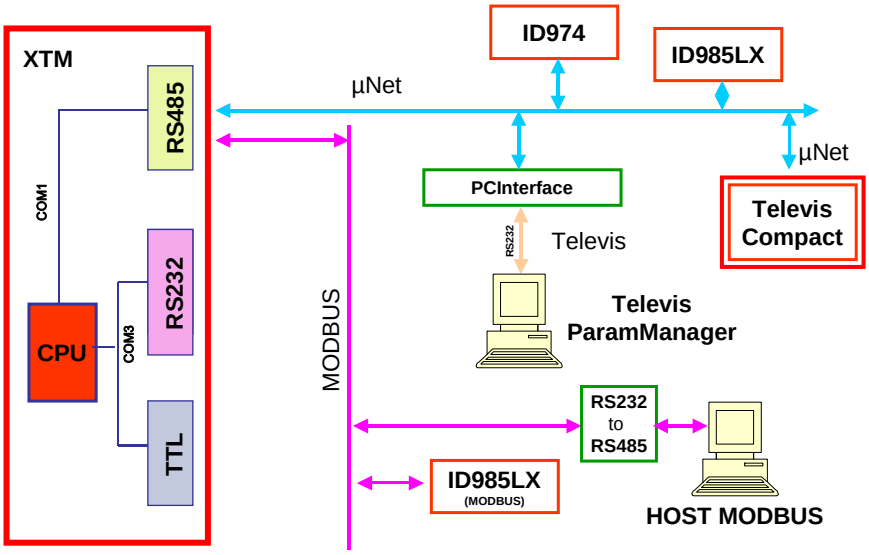
	Micronet	***ModBUS*/RTU**
***COM1*_PROTOCOLTYPE**	**2**	**3**
***COM1*_BAUD**	**hot**	**9600**
0 : 9600 b/s	x	
1 : 19200 b/s	x	
2 : 38400 b/s	x	
***COM1*_PARITY**	**Odd/even**	**hot**
0 : null		**x**
1 : odd		**x**
2 : even		**x**
***COM1*_DATA**	**8**	**8**
7 : 7 data bits		
8 : 8 data bits		
***COM1*_STOP**	**1**	**1**
1 :1 stop bit		
2 :2 stop bits		

2.3.5 Topology of local RS485

Energy XTPRO: RS485 Network connections (Debug Mode ON)



Energy XTPRO: RS485 Network connections (Debug Mode OFF)



NOTE. For Debug Mode ON/OFF refer to section ***COM1 and COM3 vs start-up without IIC card***.

2.4 "COM3" serial (RS232 / TTL)

2.4.1 Use

This serial can be used to connect the Energy XTPRO to the outside world.
This serial not only performs "slave" functions but can also spontaneously issue data packets. An example, in *MODEM control*, is its initialization string. It supports standard types of PTSN or GSM (for remote programming only) MODEM with RS232 serial connection (MODEM 485, MODEMFAX Class 1 or 2, are not implemented).

The communication speed and parity are parameter-controlled.

2.4.2 Protocols

Micronet

Micronet
To connect the instrument as SLAVE to a local network containing *Televis* as MASTER host (via PCInterface).

MODBUS

MODBUS
To connect the instrument as SLAVE
to a local network containing a *MODBUS* HOST on PC (not yet identified) as MASTER host. The *MODBUS* can be RTU (fixed baud rate of 9600 b/s) or ASCII
to a remote communication MODEM with a *MODBUS* HOST on PC (not yet identified). CAUTION, this is only possible if *MODBUS*/ASCII is used

TELEVIS

TELEVIS
To connect the instrument locally with *Televis* HOST on PC.

TELEVIS for MODEM

TELEVIS for MODEM
To connect the instrument to a remote communication MODEM with *Televis* HOST on PC.

2.4.3 COM3 PARAMETERIZATION (parameters in EEPROM highlighted)

	Micronet	*ModBUS*/RTU	*ModBUS*/ASCII	*Televis*	*Televis* MODEM
***COM3*_PROTOCOLTYPE**	**2**	**3**	**4**	**0**	**1**
***COM3*_BAUD**	**hot**	**9600**	**hot**	**hot**	**hot**
0 : 9600 b/s	x		x	x	x
1 : 19200 b/s	x		x	x	x
2 : 38400 b/s	x		x	x	x
***COM3*_PARITY**	**Odd/even**	**hot**	**hot**		
0 : null		**x**	**x**		
1 : odd		**x**	**x**		
2 : even		**x**	**x**		
***COM3*_DATA**	**8**	**8**	**hot**	**8**	**8**
7 : 7 data bits			**x**		
8 : 8 data bits			**x**		
***COM3*_STOP**	**1**	**1**	**auto (*)**	**1**	**1**
1 :1 stop bit			**x**		
2 :2 stop bits			**x**		

NOTE (*)
if (*COM3*_PROTOCOLTYPE = *Modbus*/ASCII)
then
if (*COM3*_PARITY = null and *COM3*_DATA = 7)
then *COM3*_STOP = 2
otherwise *COM3*_STOP = 1 /* (even and odd parity with data 7)
or (data 8 with any parity)*/

NOTE: If communication occurs via Modem and the protocol is *Modbus*/ASCII then operating is guaranteed for most modems if 1 stop, 8 data, parity null and 1 stop For other settings, it is important to check if the modem supports the data format.

2.4.4 MODEM control

A MODEM can be connected on *COM3* (RS232) for a fixed telephone network. GSM for remote BIOS programming.

Here is the list of parameters for the Energy XTPRO MODEM set-up:

Parameter Name	Description
PAR_BOO_BIOS_18	Modem enabling
PAR_MSG_BIOS_9	Modem initialization string (first part)
PAR_MSG_BIOS_10	Modem initialization string (continuation)
PAR_MSG_BIOS_11	Modem hangup string
PAR_ANA_BIOS_193	RS232 protocol selection
PAR_ANA_BIOS_194	Baud RS232

With PAR_BOO_BIOS_18=1 the modem is initialized when XTPRO is switched on and modem operations are enabled.

PAR_MSG_BIOS_9 and PAR_MSG_BIOS_10 are the first and second part of the modem initialization string. PAR_MSG_BIOS_10 is used only if a string that is longer than 20 characters is requested; if this is the case, make sure there are no spaces in the first part of the string (PAR_MSG_BIOS_9) since it will be truncated otherwise.
This is an example of a string that is used for most modems included those specified in the note:

AT&F&C1&D2E0X1S0=0

In many cases, we advise you to also set the value at which the connection will be forced in the string. For example, for the US ROBOTICS modem AT&F&C1&D2E0X1S0=0&N6 can be used where &N6 forces the connection at 9600 bps.

The “S0=0” part of the initialization string forces the modem not to hang up automatically. Hanging up is always effected by the XTPRO. With “S0=n” hanging up is effected by the XTPRO at the first RING.

PAR_MSG_BIOS_11 is the “Hang up” string that is used for disconnection. This is an example of a string that is used for most modems included those specified in the note:

ATH0

The parameter PAR_ANA_BIOS_193 is used to select a protocol on RS232 that supports *modem control*. More specifically:

- PAR_ANA_BIOS_193 = *Televis for Modem*;
- PAR_ANA_BIOS_193 = ASCII *MODBUS*;

Only these 2 *protocols* guarantee the *use* of NULL fixed parity (RTU *MODBUS* allows it to be selected but is a real-time control protocol).

PAR_ANA_BIOS_194 defines the communication speed between XTPRO and the modem. For a GSM modem it must always be set to 9600 bps. For an analogue modem, we recommend selecting a speed that is lower than the connection speed between 2 modems. In practice, 9600 bps is the value that guarantees connectivity in any situation. To force the US ROBOTICS modem to communicate at 9600 bps add &N6 to the initialization string.

Checking the presence of the modem is enabled when parameter PAR_BOO_BIOS_18=TRUE and it performs a periodic check on the state of the modem line and/or connection every minute.
This sequence disconnects (if still active) and reinitializes the modem.
The sequence is activated if a connection as master or slave is already active and if there is no data flow and known commands are not received. If there is no connection, the procedure is normally repeated every minute.
The sequence is interrupted if the modem does not respond to the command sent and generates a “modem HW failure” alarm [VAR_BOO_BIOS_13]; it is also interrupted if the modem responds with the string “ERROR” if the “modem SW failure” alarm is activated [VAR_BOO_BIOS_14].
In both cases, a new connection or send SMS request is accepted and the alarm is reset if completed successfully.

Note for sending SMSs

To send short text messages (SMSs) using a GSM modem make sure that the modem is already fitted with a **SIM card** with all the network services already activated and the PIN code unlocked and correct power supply and antenna. XTPRO does not check the presence/lack of signal where the modem has been installed.
To *use* a GSM modem, the parameters must be set as described above. It is also important to check that the parameter PAR_ANA_BIOS_194 is set to 9600 bps.

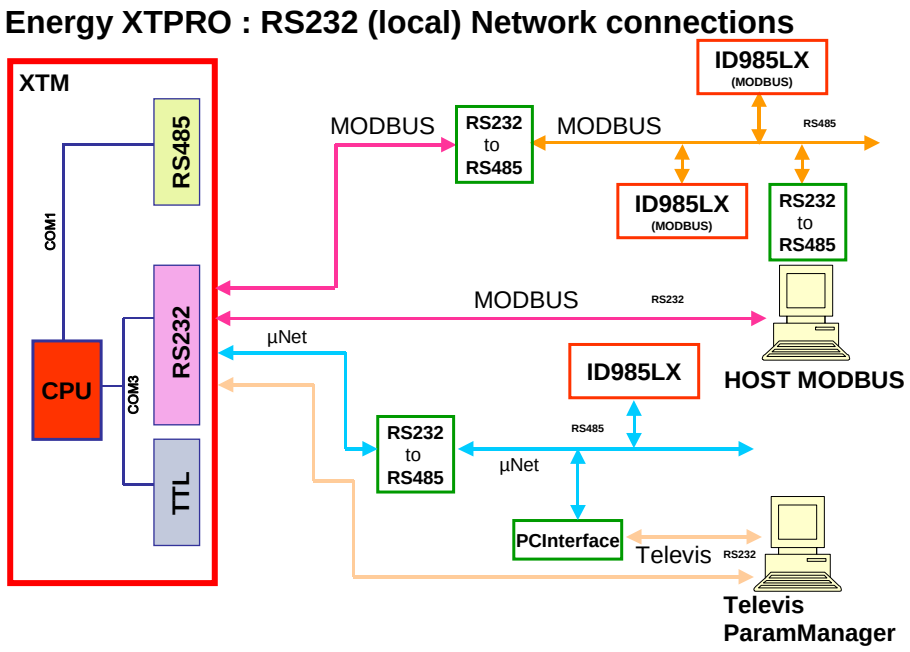
The modem call and sending of SMSs are controlled in the user application programme by way of the *Workbench* C Functions supplied by Eliwell for Energy XTPRO.

MODEM/FAX and GSM used

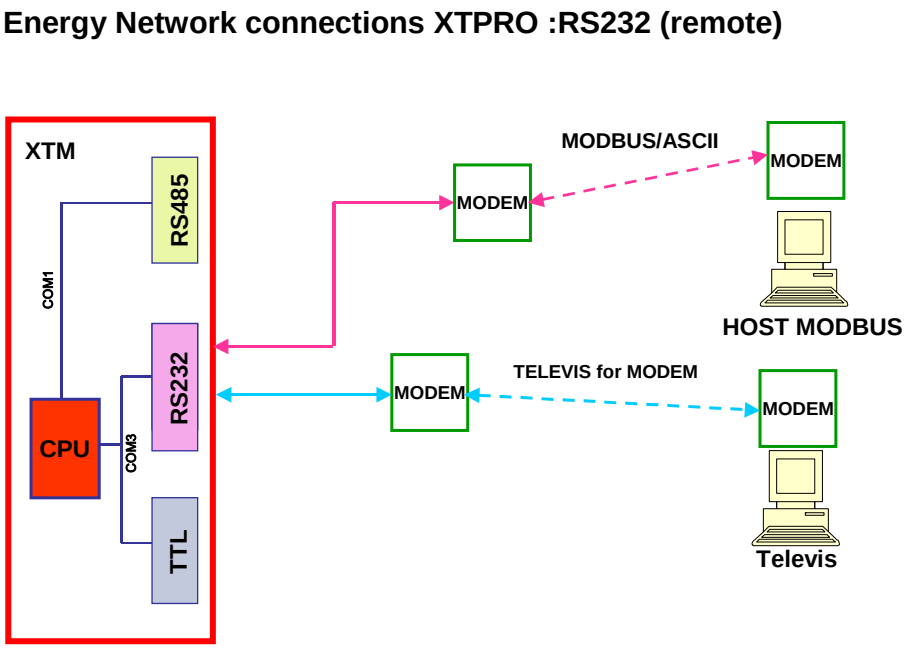
List of some MODEMS/FAXES and GSMS used:

3COM U.S.Robotics 56K Message Modem
3COM U.S.Robotics 56K FaxModem
Wavecom WMOD2 DUAL BAND MODEM (GSM modem for remote BIOS programming)

2.4.5 Topology of local RS232

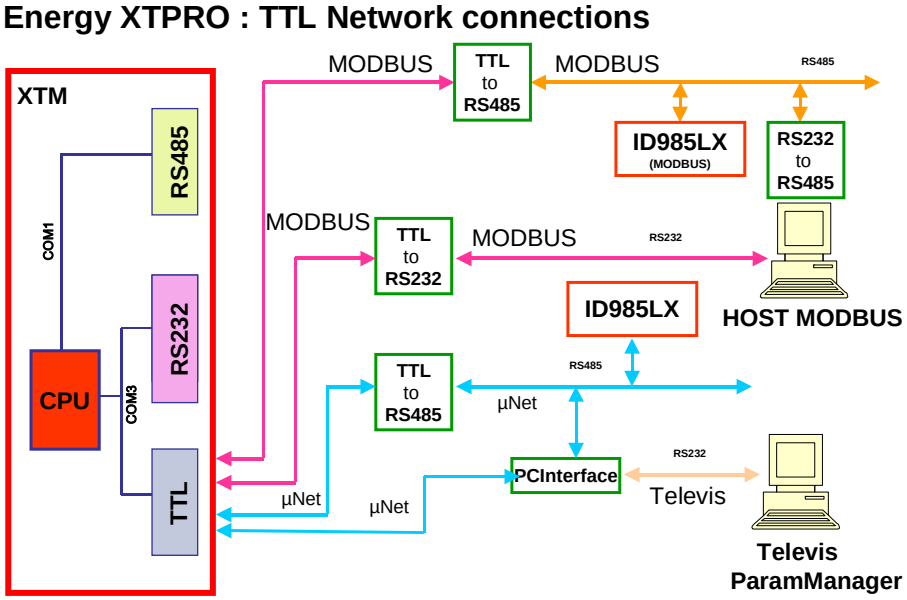


2.4.6 Topology of remote RS232

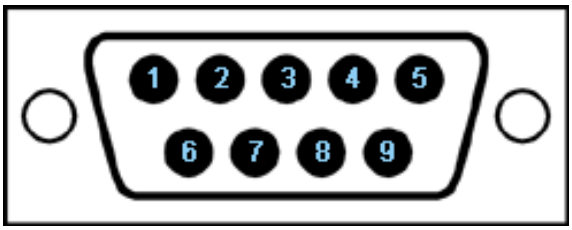


NOTE: the possibility of MODEM + *MODBUS* refers to *MODBUS*/ASCII

2.4.7 Topology of local TTL



2.4.8 SUB-D 9 MALE poles of Energy XT



SUB-D 9 MALE poles of Energy XT
(standard RS232 interface)

The pin configuration for the connection for the RS232 standard interface is shown here:

Contact N°	Code	Description
1	CD (or DCD)	Carrier Detect
2	**RxD**	**Receive Data**
3	**TxD**	**Transmit Data**
4	**DTR**	**Data Terminal ready**
5	**GND**	**Signal Ground**
6	DSR	Data Set ready
7	**RTS**	**Request to send**
8	**CTS**	**Clear to Send**
9	RI	Ring Indicator

Indicated in bold type the HW pins for the EnergyXT application.
Pin 8 CTS is not used at present and therefore the control of HW flow is not currently available

3 USER INFORMATION ON COM1 AND COM3

Both *TELEVIS* and *MODBUS* must be possible from the *Micronet* protocol:

Information	***Micronet* and *TELEVIS* command**	***MODBUS* command**
Read / write parameters and/or variables	*Resource read/write extended command*	3/16
Read / write clock	*Resource read/write extended command. Time synchronization extended command*	3/16
Read alarms	*Resource read/write extended command*	3
Block updating of state of outputs by controllers	*State read/write logical command*	16
Read/write state of digital outputs	*Digital outputs read/write logical command* *Resource read/write extended command*	3/16
Read / write state of analogue outputs	*Analogue outputs read/write logical command* *Resource read/write extended command*	3/16
Read the analogue inputs	Analogue inputs *read/write logical command*. *Resource read/write extended command*	3
Read the digital inputs	Digital inputs *read/write logical command*. *Resource read/write extended command*	3
Enable modification of COLD parameters	*State read/write logical command*	3/16
Activation of BIOS functions	*Resource read/write extended command*	3/16
Password Acknowledgement for communication enabling	*Password acknowledgement resource extended command*	**16**
FW/HW version including machine CRC (extra subfield)	**master file code command**	**43**
"Compressed" information on presence of active alarms, modified parameters..	State read/write *extended command*	**3**
Network connected device	***polling* command**	**None**

Only the information obtainable via the serial shown in **bold type** on the previous table will always be accepted by Energy XTPRO in read mode.
The other information is accepted at different acceptance levels with a password using the serial command linked to **Password Acknowledgement for communication enabling.**
More specifically:

- READ PASSWORD for enabling read-only commands
- USER PASSWORD for enabling read and write commands.
- ADMININSTRATOR PASSWORD for enabling read and write commands (not USER modifiable).

Since Energy XTPRO does not have a STATIC allocation of the resources linked to the application, this is different for every *WORKBENCH* application. The XTPRO resources CANNOT therefore by read in PHYSICAL mode.

The resources can be accessed in "mixed" LOGICAL-PHYSICAL mode according to the *WORKBENCH* application but irrespective of the BIOS (if the same BIOS version is used). For more detailed information, refer to the section on ***WORKBENCH and BIOS Resources***.

3.1 WORKBENCH and BIOS Resources

WORKBENCH and BIOS Resources

In this document, *WORKBENCH* or BIOS resources refer to the variables declared in a *WORKBENCH* project dictionary that allow a *Modbus* address.

For the *WORKBENCH* application and BIOS, it may be:

- a PARAMETER (that can be saved in the EEPROM)
- a VARIABLE (that cannot be saved in the EEPROM)

PARAMETERS and VARIABLES can be NUMERIC or MESSAGE.

If the resources belong to the BIOS their *Modbus* address will only vary according to the BIOS version used.
Users cannot modify its *Modbus* address.
If they belong to the *WORKBENCH* application, users will assign them the *Modbus* address.

3.2 WORKBENCH and BIOS resources access mode

From the table in the previous section, you can see that there are 3 categories of command:

- ***Resource read/write extended command***
- ***Extended command***
- ***Read/write logical command***

Let's examine the access to information characteristics.

3.2.1 Resource read/write extended command

This is a LOGICAL-PHYSICAL command that affects the *WORKBENCH* project resources (i.e. for the variables in a *WORKBENCH* project with a *Modbus* address).
These resources may belong to the BIOS or the *WORKBENCH* application.
If they belong to the BIOS their *Modbus* address will only vary according to the BIOS version used. Users cannot modify its *Modbus* address.

If they belong to the *WORKBENCH* application, users will assign them the *Modbus* address.
Please refer to the ***WORKBENCH and BIOS Resources*** section.

3.2.2 Extended command

This is a LOGICAL command that only depends on the BIOS and not on *WORKBENCH*.
This must not be confused with the ***Resource read/write extended command***.
It is used in this specification for Time Synchronization and File Downloading.

3.2.3 Read/write logical command

This is a LOGICAL command since it affects the logical areas. The items in each logical area are accessed via 2 coordinates (Area Index, Item Index).
The structure of the logical areas only depends on the BIOS and not on *WORKBENCH* as does the significance of the items in the areas currently in this specification version that is ABSOLUTE.

4 READ/WRITE RESOURCES WITH MODBUS COMMANDS 3 AND 16

4.1 Parameter/variable read/write extended

It reads BIOS and *WORKBENCH* parameters and variables that have a *Modbus* address.

IMPORTANT:
The write function DOESN'T responds :

- if the user writes a variable without authentication
- if the password is wrong

See **Password Acknowledgement** and/or **Enable modification of COLD parameters** commands.

4.2 Password Acknowledgement resource extended

When received from the PC is used so that the PC can tell ENERGY XTPRO what its password is and see if it is a user that is authorized to communicate.

Types of password

Types of password:

- READ PASSWORD for enabling read-only commands
- USER PASSWORD for enabling read and write commands.
- ADMININSTRATOR PASSWORD for enabling read and write commands (not USER modifiable).

The parameters associated with the *PASSWORDS* are indicated below

Parameter Name	Description
PAR_MSG_BIOS_5	Read only password: string of 10 characters
PAR_MSG_BIOS_6	User password: string of 10 characters
PAR_MSG_BIOS_7	Administrator password: string of 10 characters

IMPORTANT:
the PASSWORD is acknowledged using the write command according to the following rules:

if the PASSWORD has not been acknowledged yet then
the response will be NACK.
if not
if the PASSWORD has been acknowledged the first time then
the response will be ACK and the enables given by the PASSWORD are activated
if it is a READ_PASSWORD, the read/write of the READ PASSWORD itself will be enabled
if it is a USER_PASSWORD, the read/write of the USER PASSWORD of the READ PASSWORD will be enabled
if it is an ADMIN_PASSWORD, the read/write of the two previous *passwords* and its own will be enabled.
If
the PASSWORD has been acknowledged for a second time then
the response will be ACK and the enables given by the PASSWORD are disabled

4.3 Time synchronization extended

This is used to set seconds, minutes, hours, day of the week, day of the month, month, year,

4.4 File Download extended

This is used to download files created in the *WORKBENCH* application with these characteristics:

- They are binary files
- They will have these names: 000.txt, 001.txt, 002.txt, … , 255.txt
- They have been created by "adding" strings with a maximum of 120 characters

NOTE: if downloading is not completed, the file is left open. To close it, synchronization is requested for an inexistent file. If synchronization finds a file that has been left open by a previous download, it closes it and opens the necessary file (this may be the same one).

4.5 Digital outputs read/write logical

This is used to read and/or force the state of the Energy XTPRO relays.

NOTE: the write operation is successfully completed if the **Block updating of state of outputs by controllers** command has been previously launched and accepted.

4.6 Analogue outputs read/write logical

This is used to read and/or force the value of the Energy XTPRO analogue outputs.

NOTE: the write operation is successfully completed if the **Block updating of state of outputs by controllers** command has been previously launched and accepted.

4.7 Analogue inputs read logical

This is used to read the value of the Energy XTPRO probes.

4.8 Digital inputs read logical

This is used to read the state of the Energy XTPRO digital inputs.

4.9 State read/write logical

This is used to read and force some Energy XTPRO states or features.
They are listed below:

- Information if parameters have been modified
- Information on presence of active alarms (BIOS and USER if the USER correctly sets the BIOS variable called VAR_ANA_BIOS_3 that controls the XTK keyboard LED)
- Block updating of state of outputs by controllers. This only applies if Block Timeout has been previously set at a value than is not 0.
- Block Timeout (expressed in seconds, max. 10 minutes)
- Configuration enable from serial

An index is associated with each of these and is used in the serial command:

NP	*Modbus* Address [HEX]	*Modbus* Address [DEC]	Description of element	Value	Always readable	Readable/ Writable only after password recognition
1	8501	34049	Info if parameters have been modified	0: not modified (READ) 0: reset flag (WRITE) 1: modified (READ)	X	X
2	8502	34050	Info presence of active alarms	0: not present 1: present	X	X
3	8503	34051	Not used			
4	8504	34052	Not used			
5	8505	34053	Not used			
6	8506	34054	Update of output state blocked by regulators and input states by drivers. **NOTE (1)**	0: Unlock outputs, always carried out + reset Lock Timeout 1: Lock outputs, carried out only if Lock Timeout is different from 0	X	X
7	8507	34055	Lock Timeout	Time in seconds (max. 600 sec) NOTE: when equal to 0, any Lock is not reset	X	X
8	8508	34056	Not used			
9	8509	34057	Enabling configuration from serial. Used to write COLD parameters	0:_NON_PUOI_RICHIEDERE_INGRESSO_IN_CONFIGURAZIONE_ (READ) (ENTRY TO CONFIGURATION NOT PERMITTED (READ) 1:_NOT USED 2:_ATTENDI_PER_POTER_RICHIEDERE_INGRESSO_IN_CONFIGURAZIONE_ (READ) (WAIT TO OPEN CONFIGURATION) 3:_PUOI_RICHIEDERE_INGRESSO_IN_CONFIGURAZIONE_ (READ) (YOU CAN OPEN CONFIGURATION NOW) 4:_RICHIEDO_CONFIGURAZIONE_ (WRITE) (OPEN CONFIGURATION) 5:_SEI_IN_CONFIGURAZIONE_ (READ) (CONFIGURATION OPEN) 6:_ESCI_DALLA_CONFIGURAZIONE_ (WRITE) (EXIT CONFIGURATION) **NOTE (2)**	X	X

NOTE (1):

see also Timeout Block

NOTE (2):

If you need to access configuration from the serial port to modify COLD parameters, switch the machine off and execute an access procedure using the relevant address in the State area (see Appendix - *Troubleshooting*, Communication Test in Area 5).
You will have to enter a password (USER password is sufficient) as the procedure involves write commands.
Data is read (with *Modbus* command 3) then written (with *Modbus* command 16) to the *Modbus* address 8209 [HEX] described in the table above:

The procedure to access configuration over the serial port is described below:

1. Send read command (command 3) to address 8209
   - If the value read = 0 : configuration cannot be accessed from the serial port. **CONTACT THE ELIWELL TECHNICAL SUPPORT SERVICE TO SOLVE THE PROBLEM**
   - If the value read = 3 : execute a write command (command 16) with value = 4 to request access to configuration. Then execute a read command (command 3): if the value read = 5, then you have completed access to Configuration from Serial port.

2. To exit configuration from Serial, write value 6 (command 16).

4.10 POLLING

This MASTER command requires a generic response (ACKNOWLEDGE) to be sent by the selected SLAVE in order to establish if the device is connected to the line or not or cannot reply for some other reason (anomalies).

4.11 Master File Codes

4.11.1 FAMILY & RELEASE reading

This reads the codes in the ROM for the software family and the BIOS release version.

4.11.2 DATA EMISSIONE RELEASE reading

This reads the issue date of the BIOS release.

4.11.3 POLI (VIS/MOD) code reading

This reads the **POLI (6bit-VIS/10bit-MOD)** device code stored in the EEPROM in the PAR_ANA_BIOS_188 parameter.

4.11.4 PCH code reading

This reads the PAR_ANA_BIOS_188 parameter in the EEPROM.

4.11.5 CRC code reading

This readsthe PAR_MSG_BIOS_4 parameter.

5 MODBUS COMMANDS TO READ I/O'S SEPARATELY FROM THE APPLICATION.

5.1 Commands 3 and 16

5.1.1 Addressing sensors with Modbus command 3 or 16

The return value of each individual sensor is a number that expresses the temperature or pressure measured in tenths. The range is from –32768 to +32767. If the sensor measures –32768, an error has occurred.

E.g. if sensor AI2 reads 245, it means that it has measured 24.5°C or 24.5°F depending on sensor *parameter configuration*.

Modbus address		Sensor	Network
[DEC]	[HEX]		
33536	8300	AI1	XTM PRO
33537	8301	AI2	
33538	8302	AI3	
33539	8303	AI4	
33540	8304	AI5	
33541	8305	AI6	
33542	8306	AI7	
33543	8307	AI8	
33544	8308	AI9	
33545	8309	AI10	
33546	830A	AI11	
33547	830B	AI12	
33548	830C	AI13	
33549	830D	AI14	
33550	830E	AI15	
33551	830F	AI16	

Modbus address		Sensor	Network
[DEC]	[HEX]		
33552	8310	AI1	XTE PRO H1 XTE PRO 1
33553	8311	AI2	
33554	8312	AI3	
33555	8313	AI4	
33556	8314	AI1	XTE PRO H2 XTE PRO 2
33557	8315	AI2	
33558	8316	AI3	
33559	8317	AI4	
33560	8318	AI1	XTE PRO H3 XTE PRO 3
33561	8319	AI2	
33562	831A	AI3	
33563	831B	AI4	
33564	831C	AI1	XTE PRO H4 XTE PRO 4
33565	831D	AI2	
33566	831E	AI3	
33567	831F	AI4	

Note that if you want to read probes associated to cards not configured as present in the XTM PRO base, *Modbus* will respond with an exception command.

5.1.2 Addressing digital commands with Modbus command 3 or 16

Instructions on how to monitor digital input states in a network that is the maximum network obtainable for a current XT PRO system are provided below. The binary representation of the 16-bit registers read makes identifying the various digital inputs easy. Each bit contains the 1=excited or 0=not excited state for the relative digital input.

Network1

XTM PRO
XTE PRO H1
XTE PRO H2
XTE PRO H3
XTE PRO H4

		Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Modbus address		Digital inputs															
[DEC]	[HEX]																
33792	8400	IDL16	IDL15	IDL14	IDL13	IDL12	IDL11	IDL10	IDL9	IDL8	IDL7	IDL6	IDL5	IDL4	IDL3	IDL2	IDL1
33793	8401	IDL2	IDL1	IDL8	IDL7	IDL6	IDL5	IDL4	IDL3	IDL2	IDL1	IDL22	IDL21	IDL20	IDL19	IDL18	IDL17
33794	8402	IDL2	IDL1	IDL8	IDL7	IDL6	IDL5	IDL4	IDL3	IDL2	IDL1	IDL8	IDL7	IDL6	IDL5	IDL4	IDL3
33795	8403	0	0	0	0	0	0	0	0	0	0	IDL8	IDL7	IDL6	IDL5	IDL4	IDL3

Network2 is a lesser network than Network1 but helps you to understand how to identify inputs in the event of a "mixed" network”.

Network2

XTM PRO
XTE PRO 1
XTE PRO 2
XTE PRO 3
XTE PRO 4

Modbus address		Digital inputs															
[DEC]	[HEX]																
33792	8400	IDL16	IDL15	IDL14	IDL13	IDL12	IDL11	IDL10	IDL9	IDL8	IDL7	IDL6	IDL5	IDL4	IDL3	IDL2	IDL1
33793	8401	IDL2	IDL1	IDL4	IDL3	IDL2	IDL1	IDL4	IDL3	IDL2	IDL1	IDL22	IDL21	IDL20	IDL19	IDL18	IDL17
33794	8402	0	0	0	0	0	0	0	0	0	0	IDL4	IDL3	IDL2	IDL1	IDL4	IDL3

Note that if you want to read digital inputs associated to cards not configured as present in the XTM PRO base, *Modbus* will respond with an exception command if the digital inputs concerned are the only ones read in the register. If there are digital inputs in the register associated to cards configured as present, the state of the "non-configured" inputs will be 0.

5.1.3 Addressing analogue outputs with Modbus command 3 or 16

Instructions on how to monitor analogue output values in a network that is the maximum network obtainable for a current XT PRO system are provided below. The representation with the most significant byte (Bit15...Bit8) and least significant (Bit7...Bit0) of the 16-bit registers read makes identifying the various analogue outputs easy. Each byte contains the percentage value of the analogue output indicated. Therefore the value of the two bytes forming each register can go from 0 (corresponding to 0% of the analogue output) to 100 (corresponding to 100% of the analogue output).

Network1

XTM PRO
XTE PRO H1
XTE PRO H2
XTE PRO H3
XTE PRO H4

		Bit15...Bit8	Bit7... Bit0
Modbus **address**		**Analogue outputs**	
[DEC]	**[HEX]**		
35072	8900	AO2	AO1
35073	8901	AO4	AO3
35074	8902	AO2	AO1
35075	8903	AO2	AO1
35076	8904	AO2	AO1
35077	8905	AO2	AO1

Network2 is a lesser network than Network1 but helps you to understand how to identify inputs in the event of a "mixed" network".

Network2

XTM PRO
XTE PRO 1
XTE PRO 2
XTE PRO 3
XTE PRO 4

Modbus **address**		Analogue outputs	
[DEC]	[HEX]		
35072	8900	AO2	AO1
35073	8901	AO4	AO3

Note that if you want to read analogue outputs associated to cards not configured as present in the XTM PRO base, *Modbus* will respond with an exception command.

5.1.4 Addressing digital outputs with Modbus command 3 or 16

Instructions on how to monitor digital output states in a network that is the maximum network obtainable for a current XT PRO system are provided below. The binary representation of the 16-bit registers read makes identifying the various relays easy. Each bit contains the ON=1 or OFF=0 state of the respective digital output.

Max Network

XTM PRO
XTE PRO H1
XTE PRO H2
XTE PRO H3
XTE PRO H4

		Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Modbus **address**		**Digital outputs**															
[DEC]	**[HEX]**																
34816	8800	NO16	NO15	NO14	NO13	NO12	NO11	NO10	NO9	NO8	NO7	NO6	NO5	NO4	NO3	NO2	NO1
34817	8801	NO12	NO11	NO10	NO9	NO8	NO7	NO6	NO5	NO4	NO3	NO2	NO1	NO20	NO19	NO18	NO17
34818	8802	NO13	NO12	NO11	NO10	NO9	NO8	NO7	NO6	NO5	NO4	NO3	NO2	NO1	NO15	NO14	NO13
34819	8803	NO14	NO13	NO12	NO11	NO10	NO9	NO8	NO7	NO6	NO5	NO4	NO3	NO2	NO1	NO15	NO14
34820	8804	NO15	NO14	NO13	NO12	NO11	NO10	NO9	NO8	NO7	NO6	NO5	NO4	NO3	NO2	NO1	NO15

The Mixed Network is obtained through a combination of XT and expansions with less inputs and outputs than the MaxNetwork, but which helps you to understand how to identify digital outputs in the event of a "mixed" network".

Mixed Network

XTM PRO
XTE PRO 1
XTE PRO 2
XTE PRO 3
XTE PRO 4

Modbus **address**		**Digital outputs**															
[DEC]	**[HEX]**																
34816	8800	NO16	NO15	NO14	NO13	NO12	NO11	NO10	NO9	NO8	NO7	NO6	NO5	NO4	NO3	NO2	NO1
34817	8801	NO3	NO2	NO1	NO9	NO8	NO7	NO6	NO5	NO4	NO3	NO2	NO1	NO20	NO19	NO18	NO17
34818	8802	NO1	NO9	NO8	NO7	NO6	NO5	NO4	NO3	NO2	NO1	NO9	NO8	NO7	NO6	NO5	NO4
34819	8803	0	0	0	0	0	0	0	0	NO9	NO8	NO7	NO6	NO5	NO4	NO3	NO2

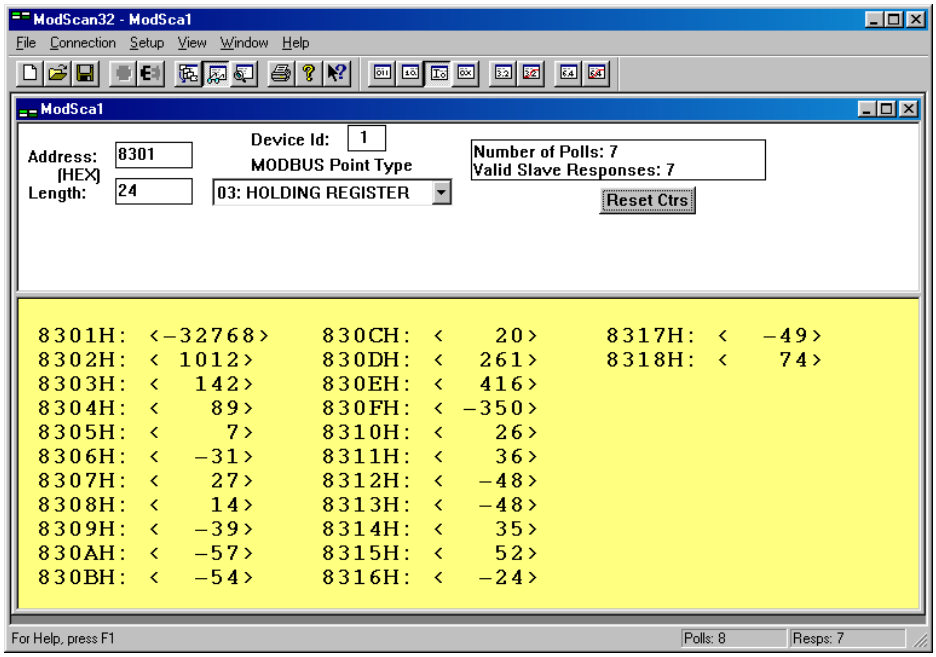
Note that if you want to read digital outputs associated to cards not configured as present in the XTM PRO base, *Modbus* will respond with an exception command if the digital outputs concerned are the only ones read in the register. If there are digital outputs in the register associated with cards configured as present, the state of the "non-configured" outputs will be 0.

6 STRUCTURE OF MODBUS COMMANDS - EXAMPLES

6.1.1 Examples with Modscan32 for Command 3 with bit16=1

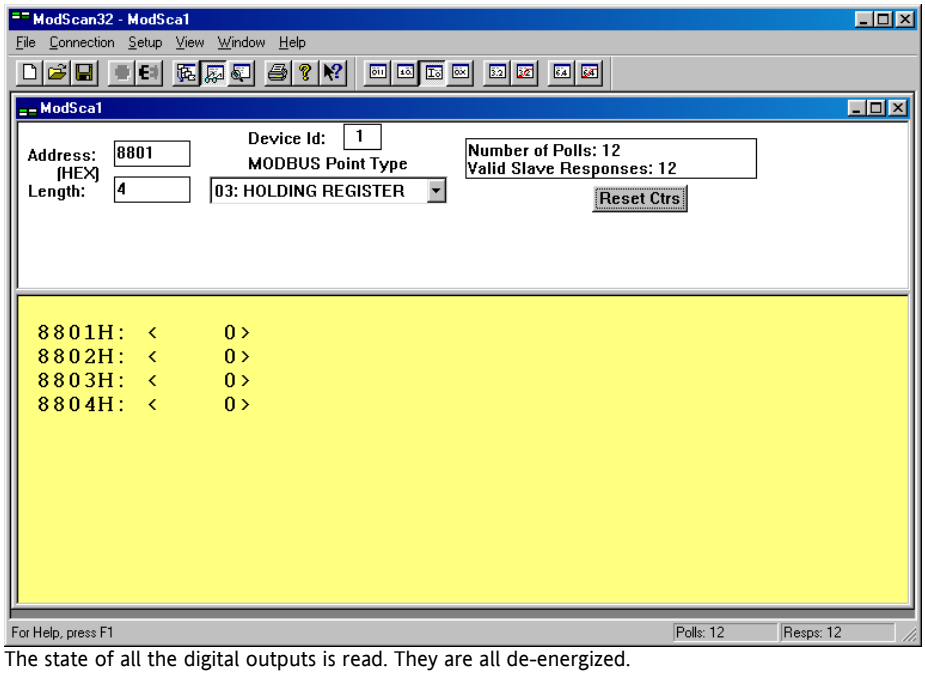
The examples have been created using an Energy XTMH with two XTEH external expansions. Note that Modscan32 in the Address field contains the address that you wish to read+1. This means that the Modscan address 0x8301 corresponds to reading the logical area 3 starting with item 0 (i.e. the first item).

Reading probes logical area



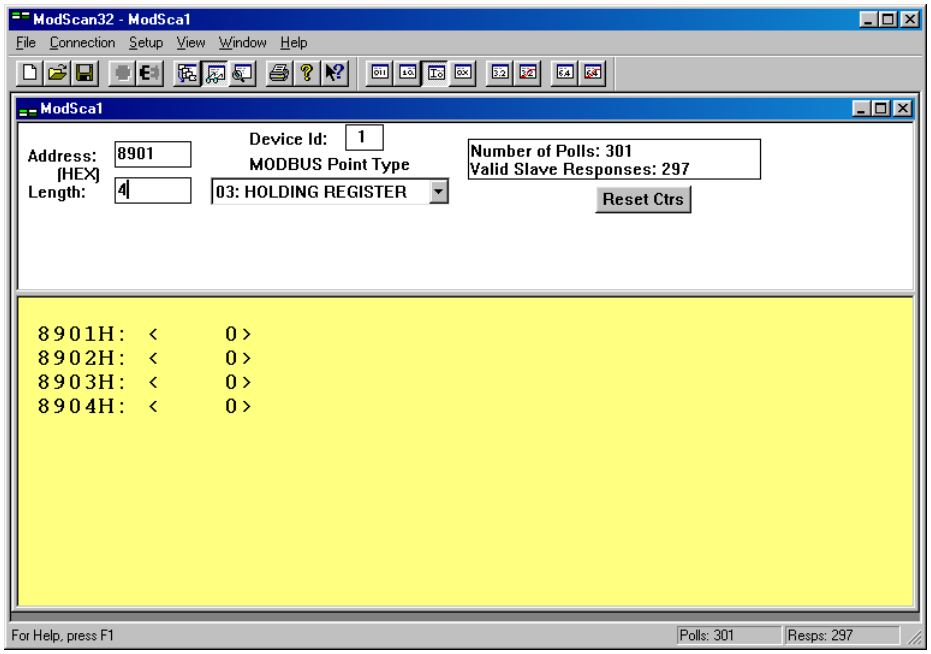
Probes from AI1 to AI16 of the XTMH base (from 8301H to 8310H) are read + the four probes on the first external expansion (from 8311H to 8314H) + the four probes on the second external expansion module (from 8315H to 8318H). Note that AI1 on the XTMH is faulty (8000H = -32768). All values are expressed in tenths.

Reading Digital Outputs logical area



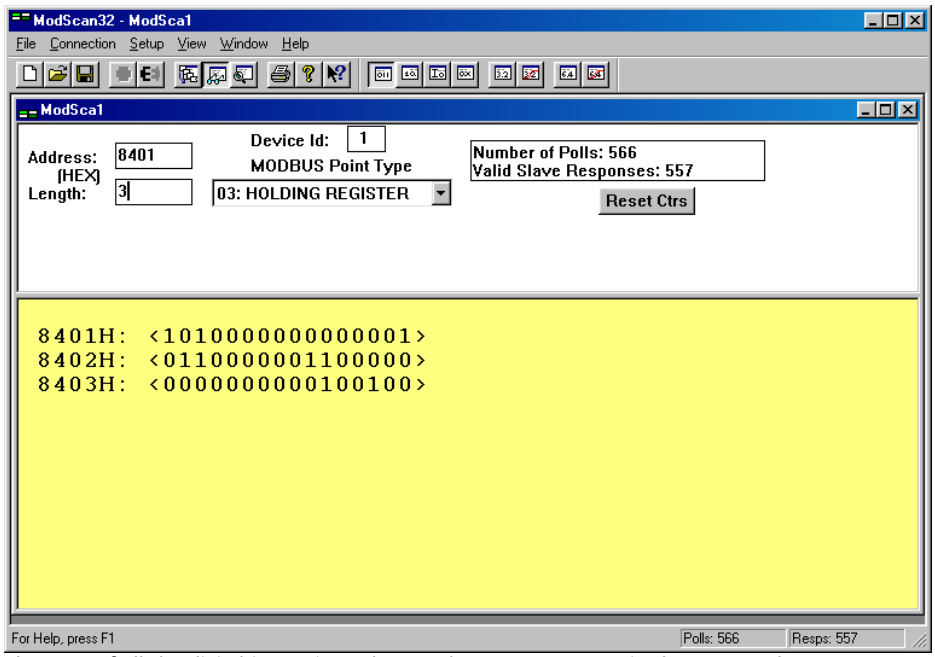
The state of all the digital outputs is read. They are all de-energized.

Reading Analogue Outputs logical area



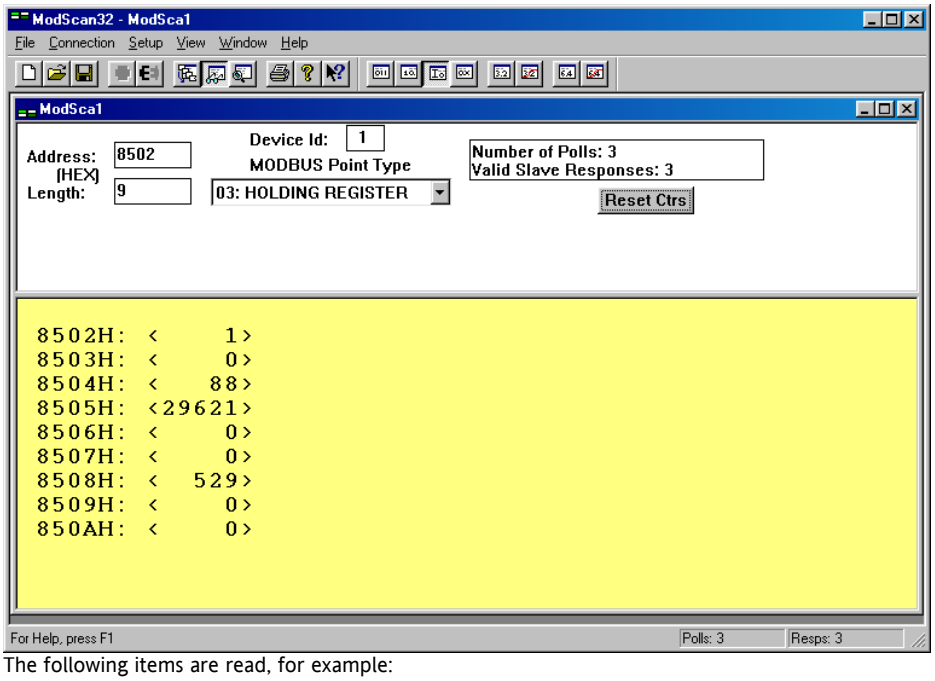
The value of all 8 AOs equalling 0 % is read.

Reading Digital Inputs logical area



The state of all the digital inputs is read. Note that some are energized. For example IDL1, IDL14, IDL16 and IDL22 on the XTMH base + IDL1 and IDL8 on the first external expansion module + IDL1, IDL5 e IDL8 on the second external expansion module.

Reading State logical area

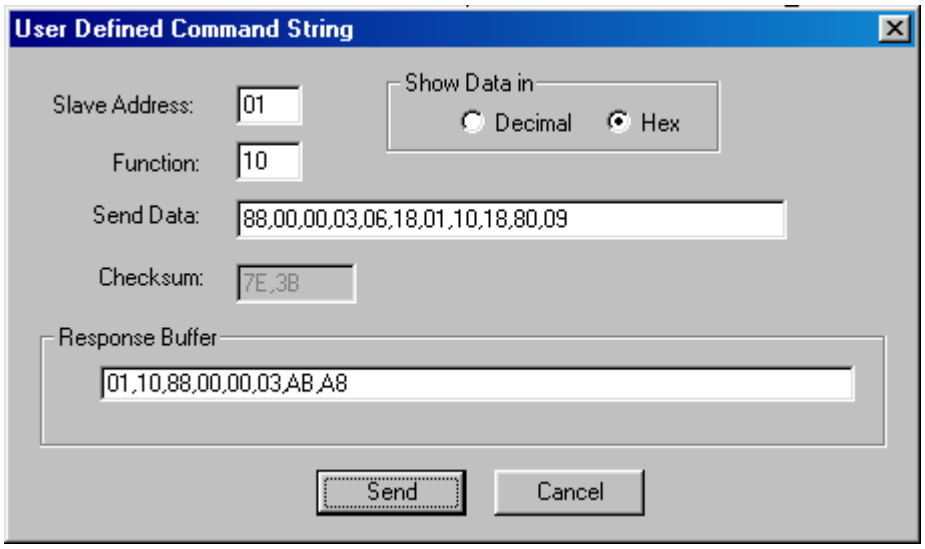


The following items are read, for example:

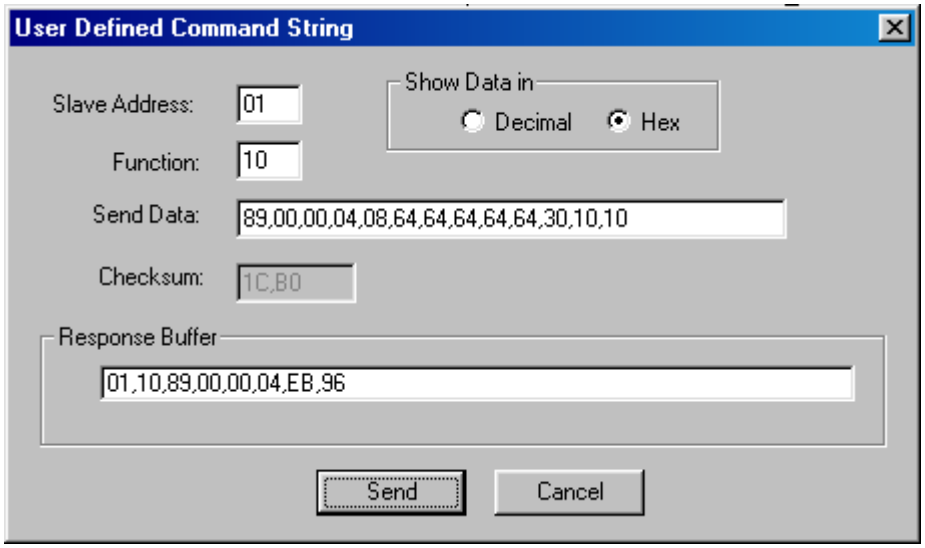
NP	Description of item
1	Info if parameters have been modified
2	Info on presence of active alarms
3	Not used: insignificant value
4	Not used: insignificant value
5	Not used: insignificant value
6	Block updating of state of outputs by controllers.
7	Block Timeout
8	Not used
9	Configuration enable from serial
10	Not used

6.1.2 Examples with Modscan32 for Command 16 with bit16=1

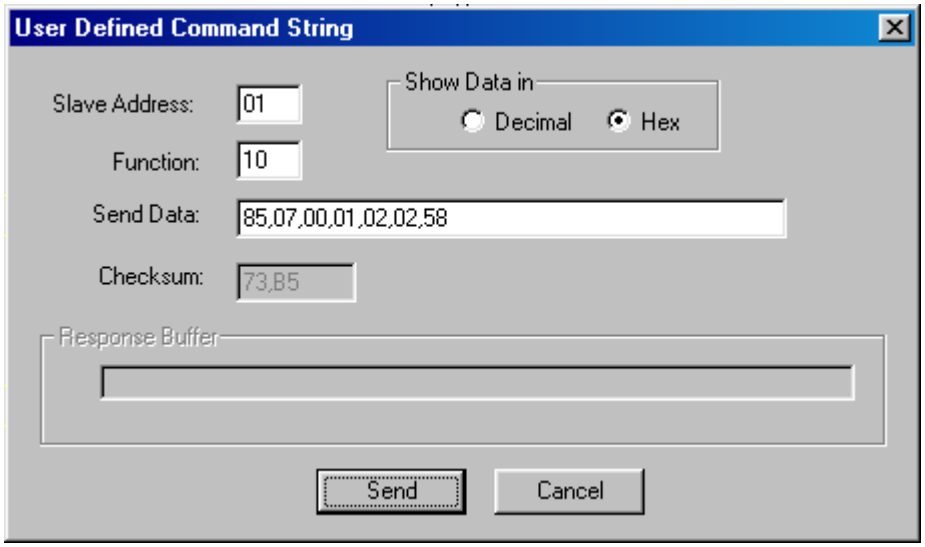
Reading NO on Digital Outputs logical area



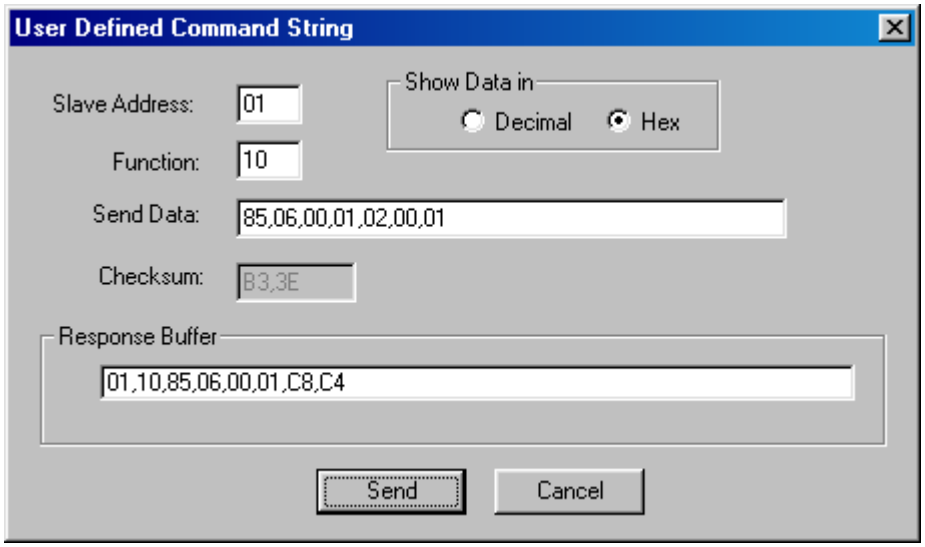
Reading AO on Digital Outputs logical area



Writing item 7 of States area to 600 (Block Timeout)



Writing item 6 of States area to 1 (Block updating of state of outputs by controllers)

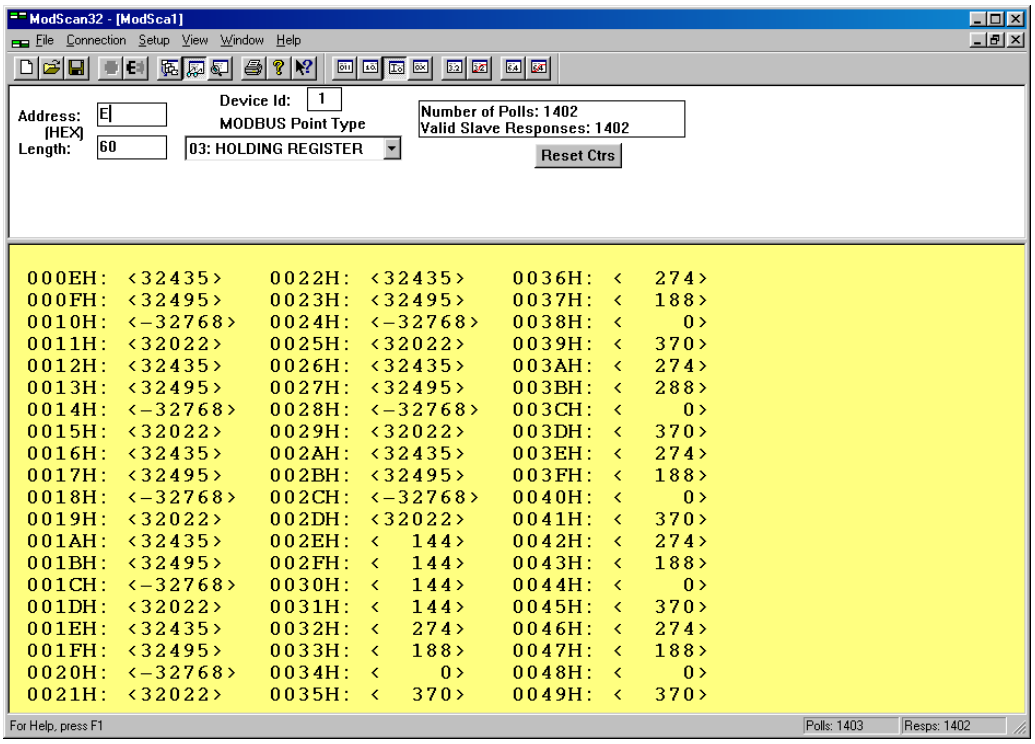


6.1.3 Examples with Modscan32 for Command 3 with bit16=0

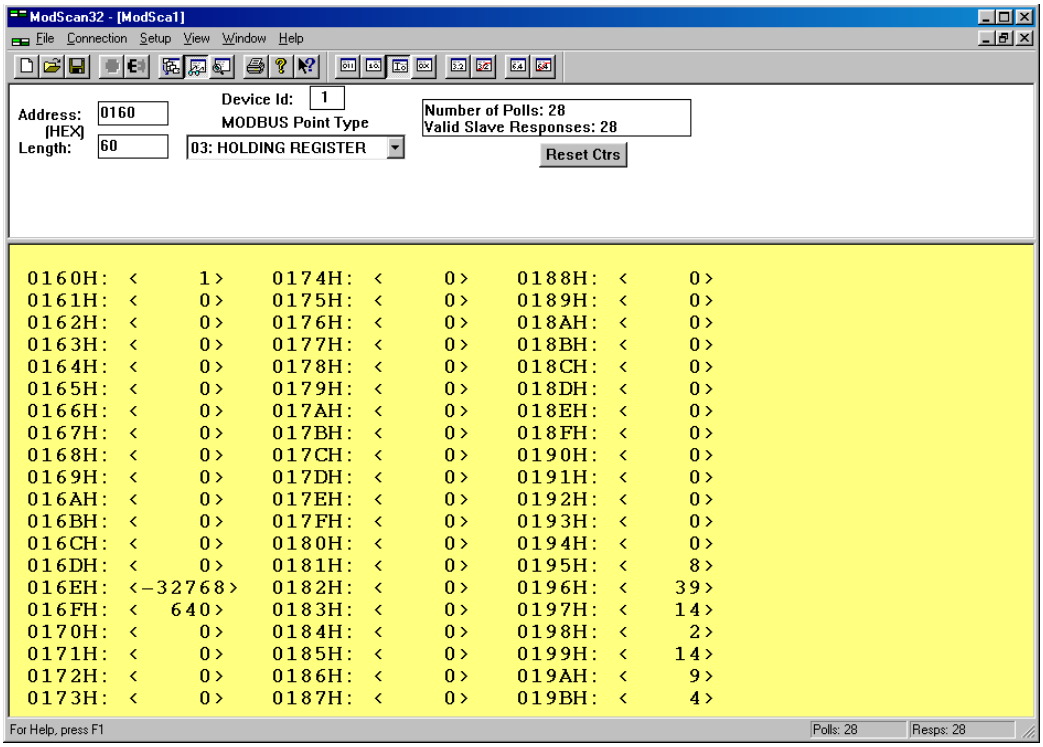
The examples have been created using an Energy XTMH with two XTEH external expansions.

Note that Modscan32 in the Address field contains the address that you wish to read+1. This means that the Modscan address 0x000E corresponds to reading the Modscan address 0x000D on the Energy XTPRO.

Reading numeric parameters

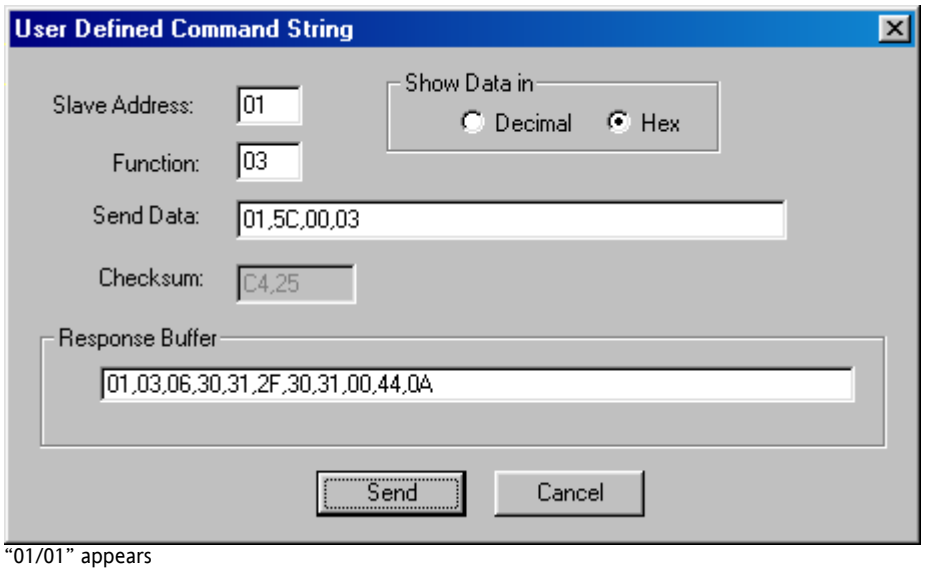


Reading numeric variables



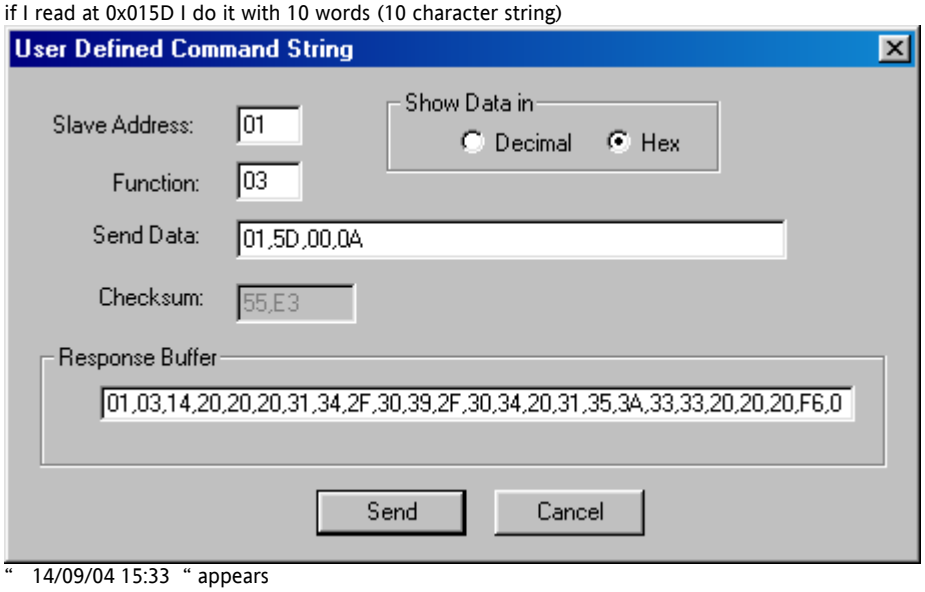
Reading string variables

if I read at 0x015C I do it with 3 words (5 character string)



"01/01" appears

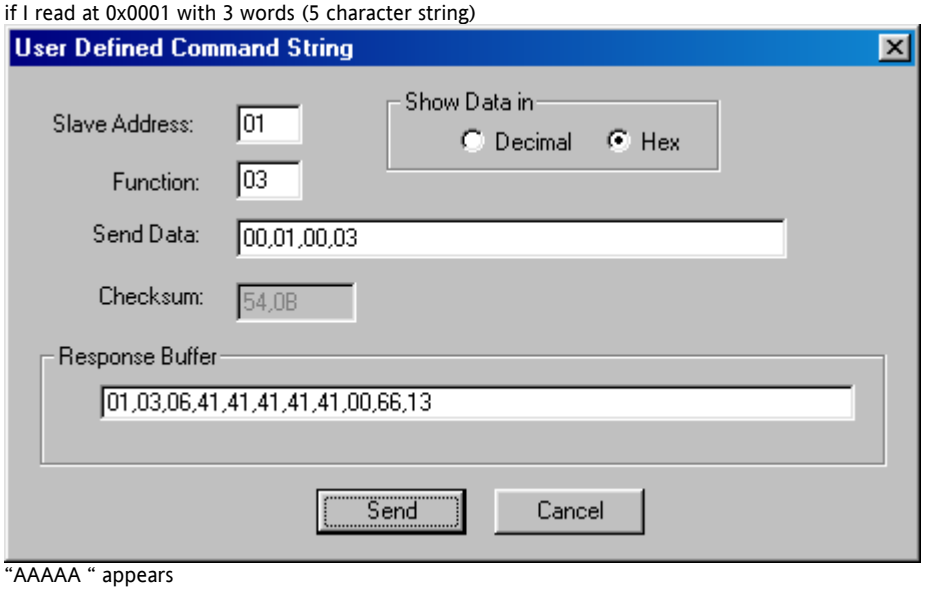
if I read at 0x015D I do it with 10 words (10 character string)



" 14/09/04 15:33 " appears

Reading string parameters

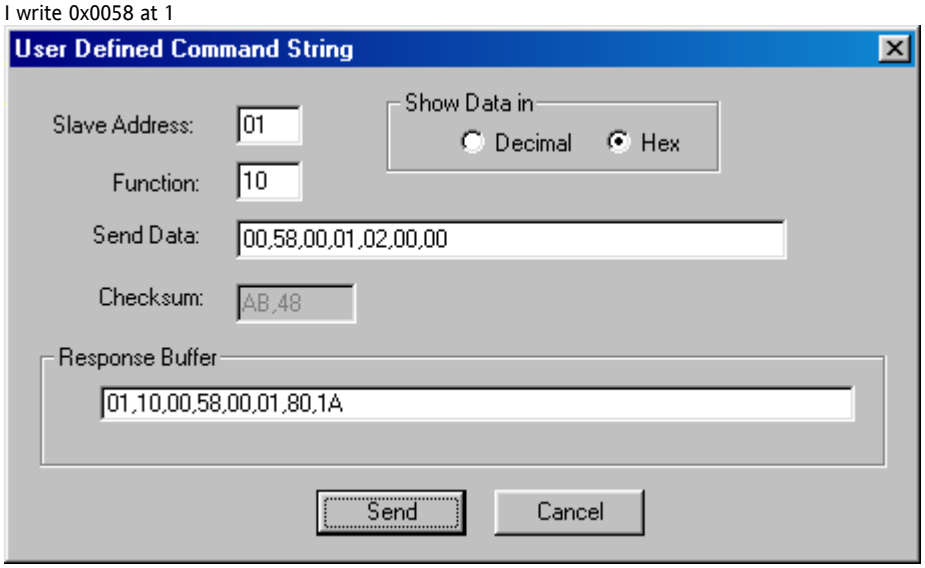
if I read at 0x0001 with 3 words (5 character string)



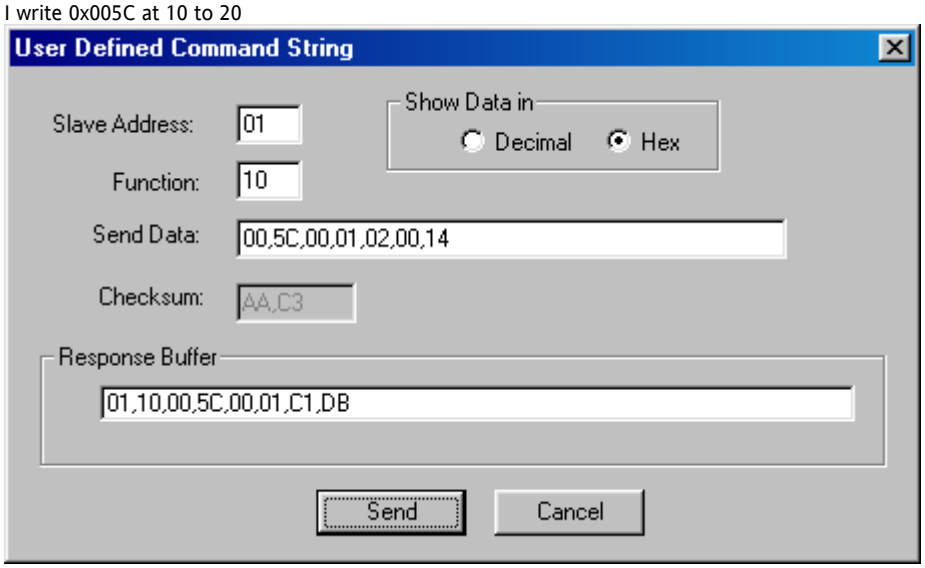
"AAAAA " appears

6.1.4 Examples with Modscan32 for Command 16 with bit16=0

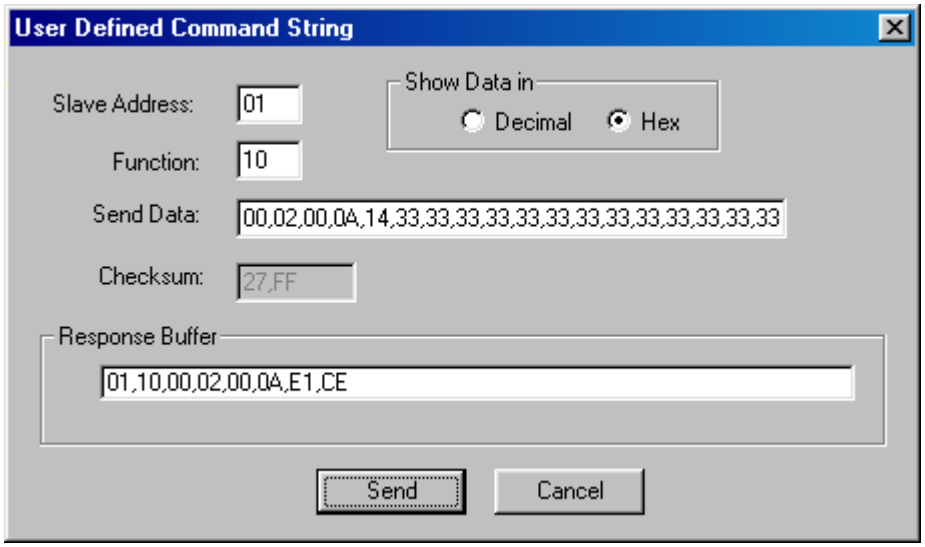
I write 0x0058 at 1

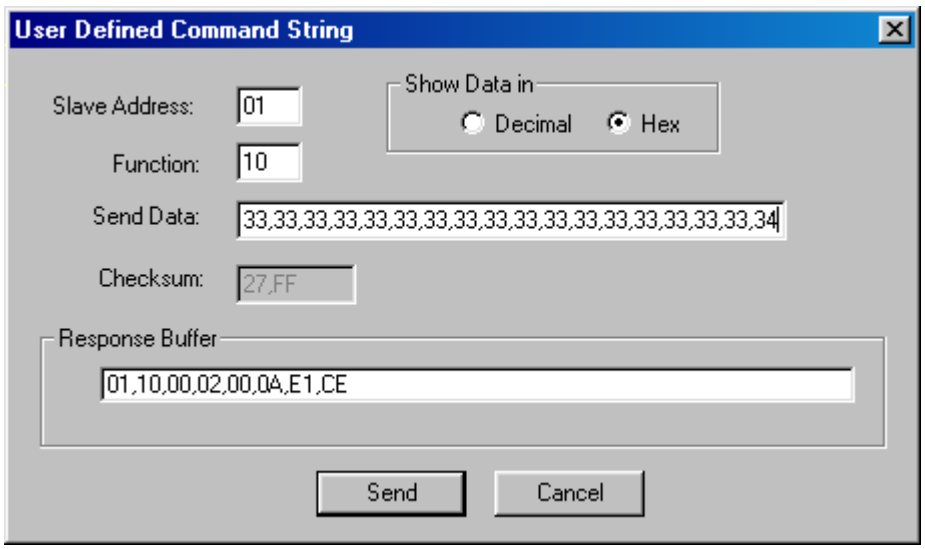


I write 0x005C at 10 to 20

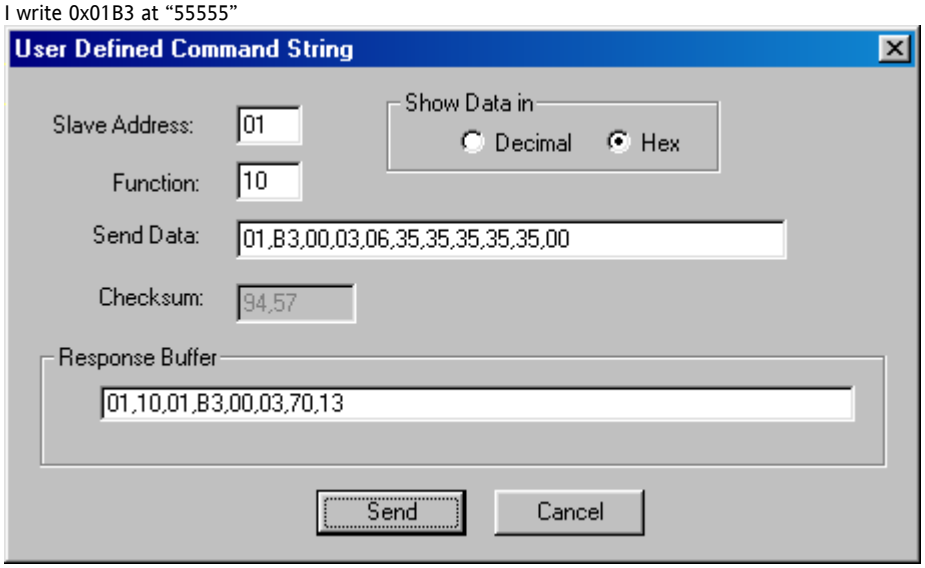


I write 0x002 at “3333333333333333334”





I write 0x01B3 at “55555”



6.2 Command 20

The **20 Read File Record** command is used to download files created in the *WORKBENCH* application with these characteristics:

- They are binary files
- They will have these names: 000.txt, 001.txt, 002.txt, … , 255.txt
- They have been created by "adding" strings with a maximum of 120 characters

The command has been implemented so that data can only be requested <u>for one file at a time</u> (data cannot be requested for several files in the same frame).

<u>Format of request PDU</u>

Function code	1 Byte	0x14	Fixed value equal to command code
Byte Count	1 Byte	7	Fixed value
Sub-Req x Reference Type	1 Byte	6	Fixed value
Sub-Req x File Number	2 Bytes	N_FILE	File name = number of file that you want to download. E.g.: 023.txt -> 23dec therefore N_FILE = 17h
Sub-Req x Record Number	2 Bytes	0…N_REC	File record index: 0=give me the first (Synchronization) otherwise=give me the next ones
Sub-Req x Record Length	2 Bytes	0x3D	Fixed value equal to size in words of record

NOTE:

The first frame must have the Record Number at 0 so that the Energy XTPRO can be positioned, i.e. synchronized on the first record of the file. Subsequent frames will have a Record Number that is not 0 (it may be any value except 0: to clarify the situation, reading successive records in the same file could be an ascending number) to indicate that resynchronization is not required and other records in the file can be sent. If the Record Number is placed at 0 during downloading, the XTPRO repositions itself on the first record of the selected file.

NOTE:

if downloading is not completed, the file is left open. To close it, synchronization is requested for an inexistent file. If synchronization finds a file that has been left open by a previous download, it closes it and opens the necessary file (this may be the same one).

<u>Format of response PDU</u>

Function code	1 Byte	0x14	Fixed value equal to command code
Resp. data length	1 Byte	0x7C	Fixed value
Sub-Req x file resp. length	1 Byte	0x7B	Fixed value
Sub-Req x Reference Type	1 Bytes	6	Fixed value
Sub-Req x Record Data	122 Bytes		Record data indexed by request command

<u>Structure of data contained in Sub-Req x 122 byte long Record Data:</u>

first byte	LEN	Number of bytes actually belonging to the downloaded record counted from third byte. Successive data has no significance and must not therefore be considered
second byte	CODE	0x00=End of file download 0x01=Good data, you can continue downloading 0xFB=No synchronization 0xFC=Buffer tx not available (downloading on other serial) 0xFD=Item reading error 0xFE=file does not exist or opening error 0xFF=file empty
third byte		first data point of record
:		
:		
one hundred and twenty second byte		one hundred and twenty first data point of record

NOTE:

The response to a request for an item in the collection without initial synchronization has the value 0xFB in the CODE field and length LEN =0.
The response to a request for an item in the collection when a collection is being downloaded from the other serial has the value 0xFC in the CODE field and length LEN =0.
To signal errors in reading a record in the collection, the response will have these values in the CODE field:

- 0xFD=Item reading error
- 0xFE=Collection does not exist
- 0xFF=collection is empty

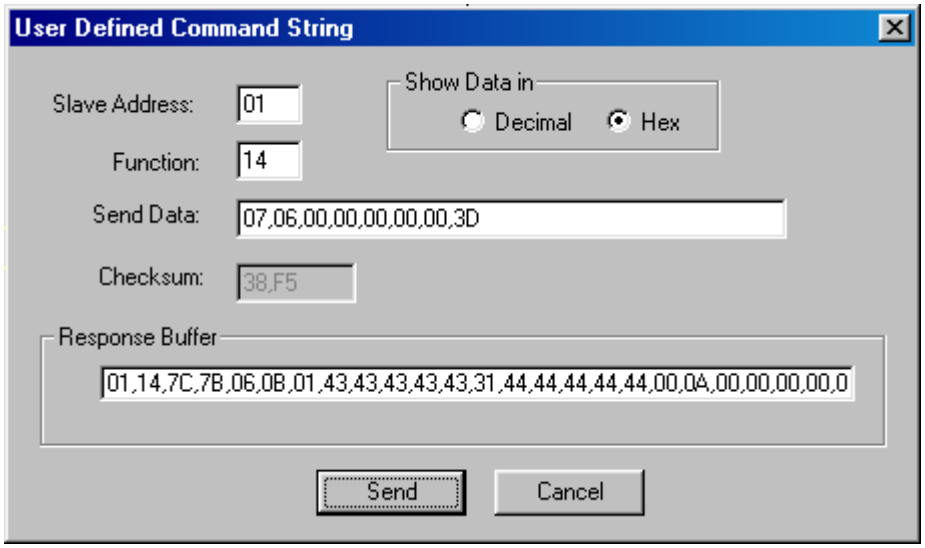
and length LEN =0.

During downloading of data CODE=0 and the length LEN is the exact number of useful data in the data field. At the end of downloading the field CODE=0 and LEN =0.
From this point onwards, the lack of synchronization will be signalled at each request.

When CODE=0xFC (buffer already used by the other serial) downloading of data must be completed with the other serial. Alternatively, to immediately reset the procedure, the serial that has begun downloading may immediately terminate it by sending the request to start downloading an inexistent file.

6.2.1 Example with Modscan32 for Command 20

The following records "CCCCC1DDDDD","DDDDD2CCCCC" and "CCCCC3DDDDD" have been written in the Energy XTPRO FileSystem in the "000.txt" file. I synchronize on the "000.txt" file and read the first record.



6.3 Command 43

The **43 Read Device Identification** command is used with the **Master File Command** of the *MICRONET* & *TELEVIS protocols*.

Function code	1 Byte	0x2B	Fixed value equal to command code
MEI Type	1 Byte	0x0E	Fixed value
Read Device ID code	1 Byte	0x04	Fixed value to request an item
Object ID	1 Byte	0,1,2,0x80	ID info code 0 = Vendor Name 1 = Product Code (PCH+POLI) 2 = MajorMinorRevision (Fam+Ver) 0x80 = private (CRC)

Format of fixed part of response PDU

Function code	1 Byte	0x2B	Fixed value equal to command code
MEI Type	1 Byte	0x0E	Fixed value
Read Device ID code	1 Byte	0x04	Fixed value to request an item
Conform. Level	1 Byte	0x81	Fixed value
More follows	1 Byte	0x00	Fixed value
Next Object ID	1 Byte	0x00	Fixed value
Number of Object ID	1 Byte	0x01	Fixed value

Format of part following fixed part of PDU responding to Object ID = 0:

Object ID	1 Byte	0	Value of Object ID
Object Length	1 Byte	9	Fixed value
Object Value	9 Byte	string	'ELIWELL ' (NOTE: there is a final space that brings the total to 9 characters)

Format of part following fixed part of PDU responding to Object ID = 1:

Object ID	1 Byte	0	Value of Object ID
Object Length	1 Byte	9	Fixed value
Object Value	9 Byte	string	ASCII string containing <PCH code in 4 bytes>1 0x5F code separation byte('-')<POLI code in 4 bytes>

Format of part following fixed part of PDU responding to Object ID = 2:

Object ID	1 Byte	0	Value of Object ID
Object Length	1 Byte	9	Fixed value
Object Value	9 Byte	string	ASCII string containing <FAM code in 4 bytes>1 0x5F code separation byte('-')<VER code in 4 bytes>

Format of part following fixed part of PDU responding to Object ID = 0x80:

Object ID	1 Byte	0	Value of Object ID
Object Length	1 Byte	20	Fixed value
Object Value	20 Byte	string	ASCII string containing the CRC

Example:

CMD_**43** obj **0**	Reading master file information: obj 0 vendor: "**ELIWELL** "
CMD_**43** obj **1**	Reading master file information: obj 1 PCH_POLI_2: "**0000_0000**" -> 10_401 -> 16_1025 -> **PCH** =16 and **POLI** = 1025
CMD_**43** obj **2**	Reading master file information: obj 1 MSK_REL: "**00D5_0006**" -> D5_06 -> 213_06 -> **MSK** =213 and **REL** = 6

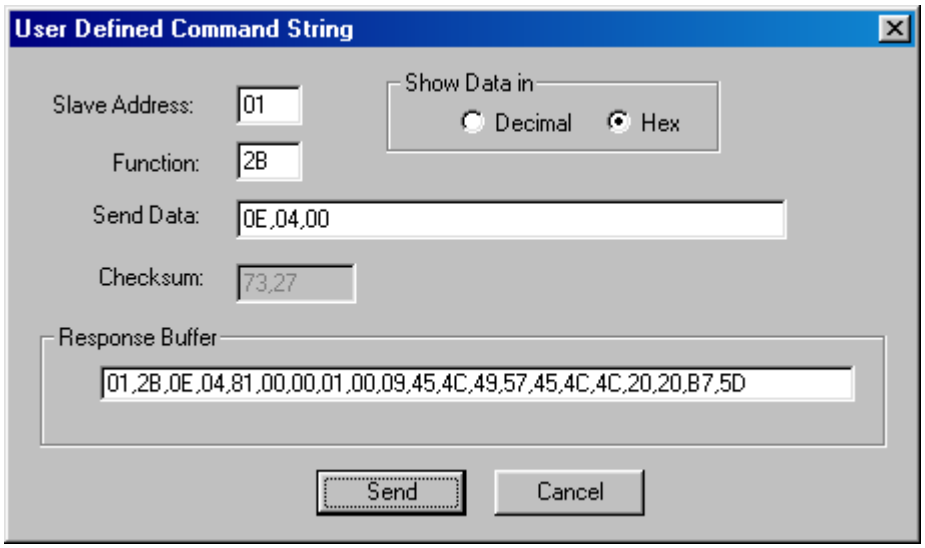
where:

- the OBJ 0: is a constant ASCII STRING for the company, we have decided to indicate "ELIWELL" excluding superscripts;
- the OBJ 1: is an ASCII STRING consisting of: the ASCII representation of the hexadecimal value of the PCH parameter formatted on 4 digits, the ASCII underscore symbol "_" excluding superscripts, the ASCII representation of the hexadecimal value of the POLI parameter formatted on 4 digits;

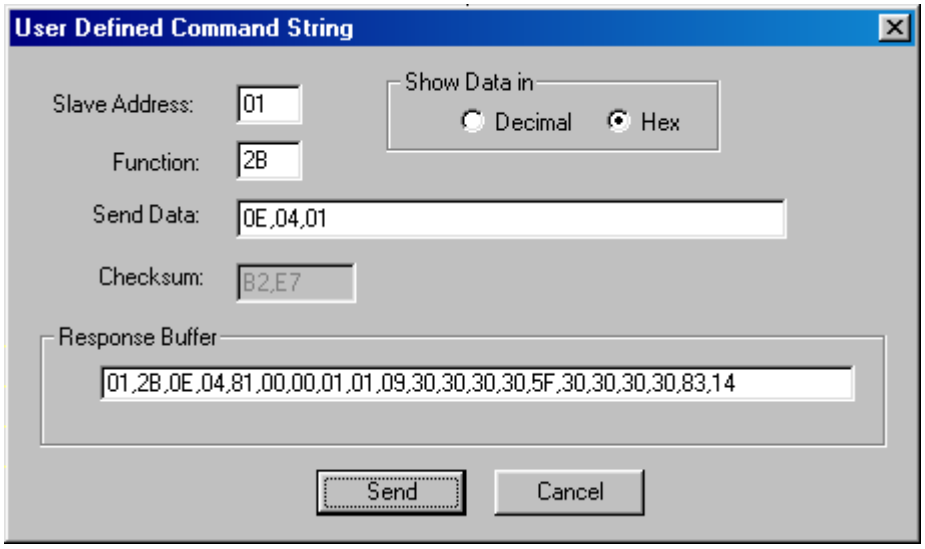
- the OBJ 2: is an ASCII STRING consisting of: the ASCII representation of the hexadecimal value of the MSK parameter formatted on 4 digits, the ASCII underscore symbol "_" excluding superscripts, the ASCII representation of the hexadecimal value of the REL value formatted on 4 digits;

6.3.1 Examples with Modscan32

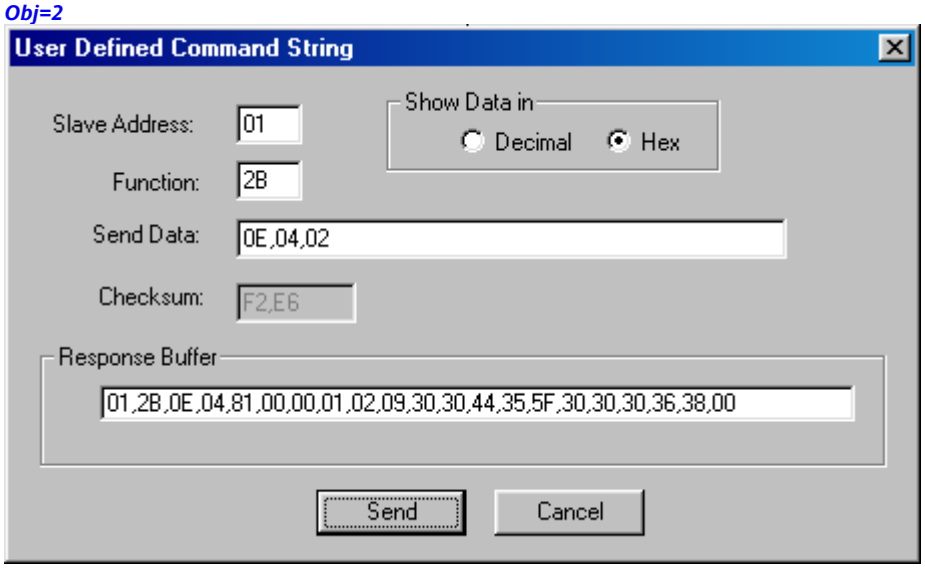
Obj=0 *Obj=0*



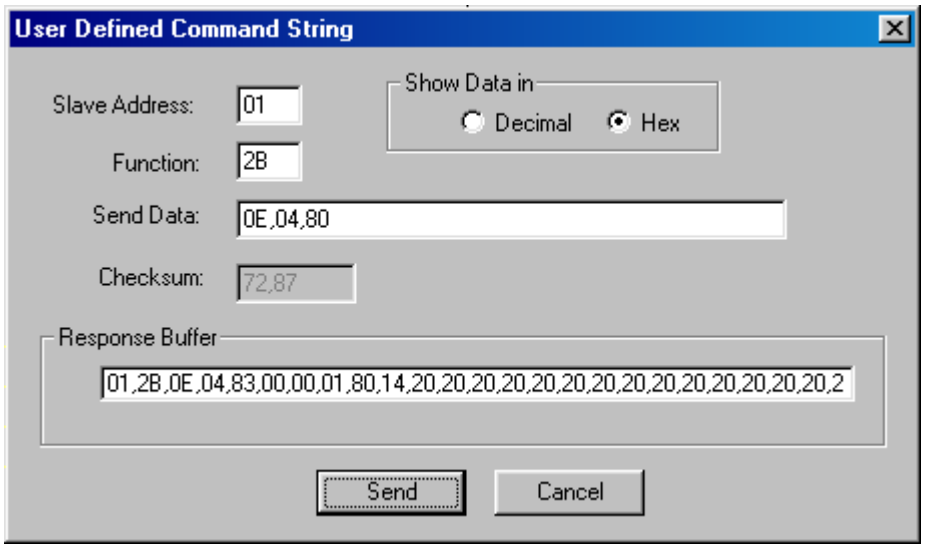
Obj=1 *Obj=1*

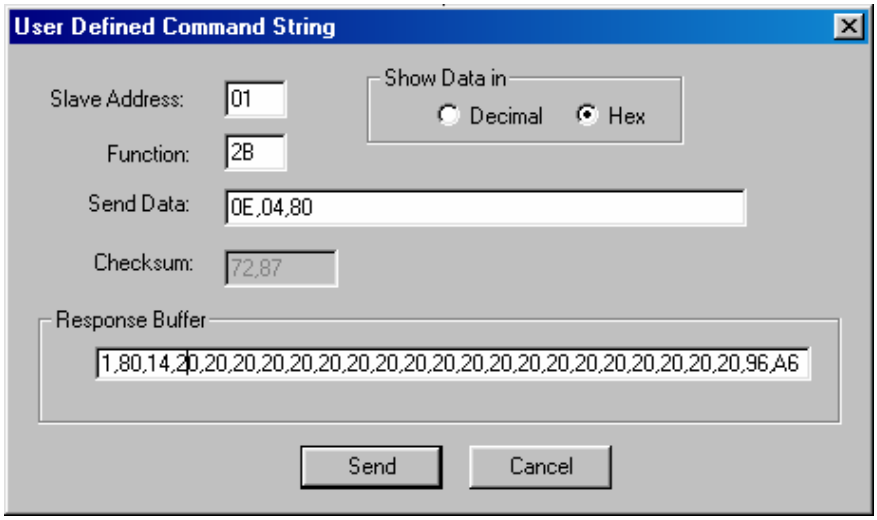


Obj=2 *Obj=2*



Obj=0x80 *Obj=0x80*

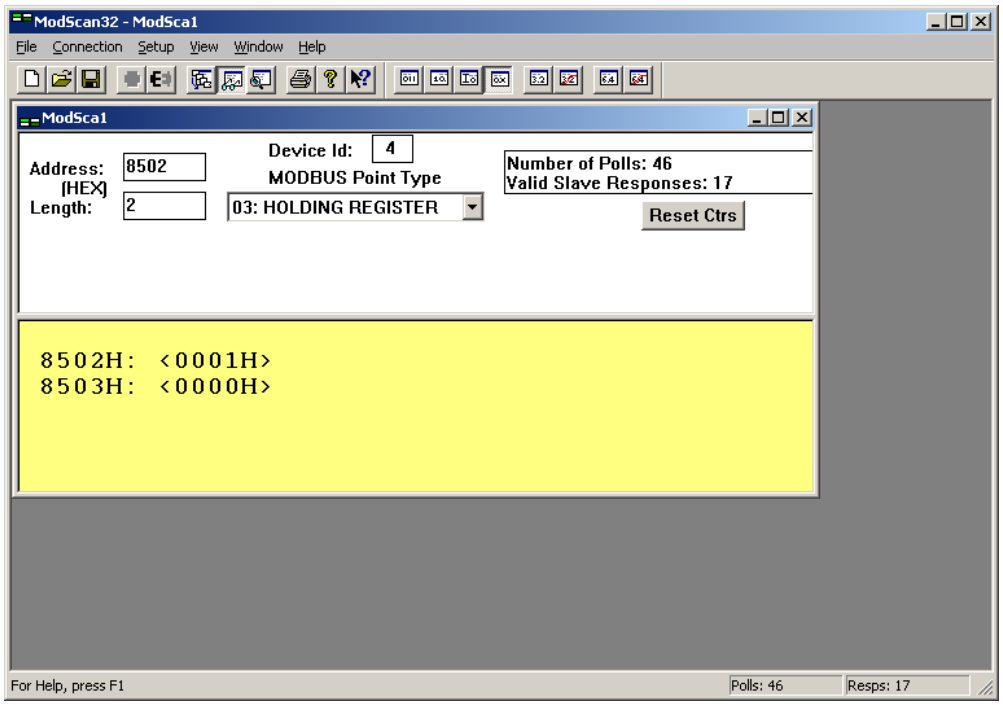


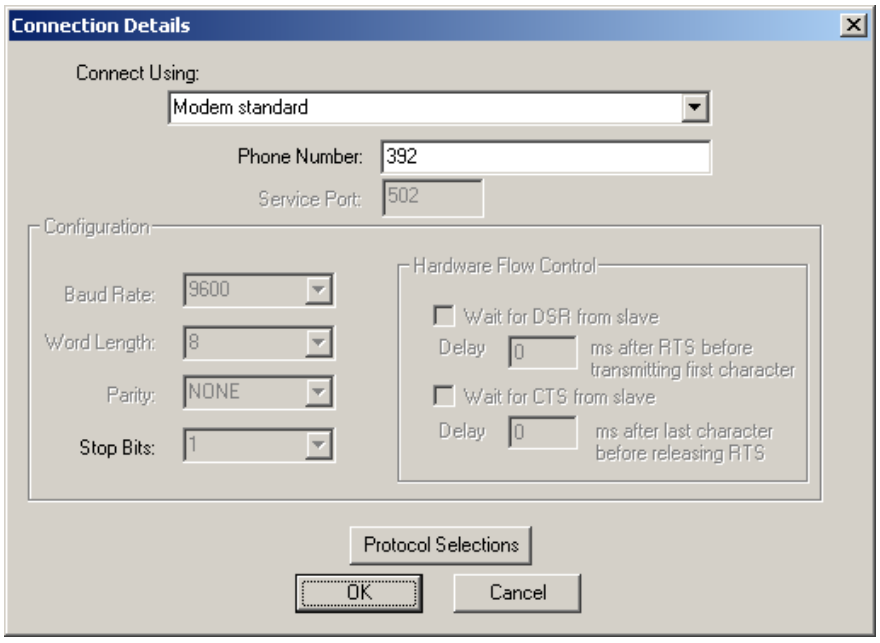


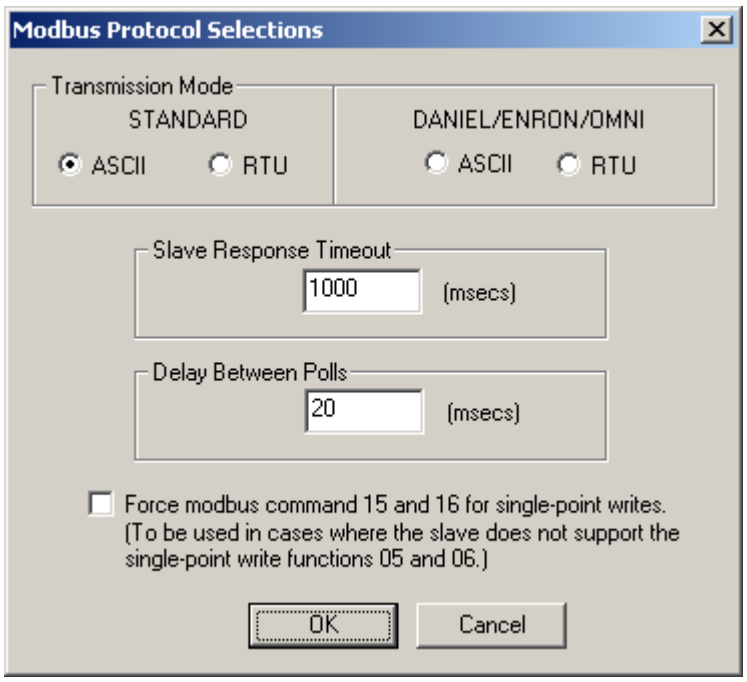
6.4 ModScan32 connected to a modem via RS-232

ModScan32 connected to Trust 56k v.92 External Modem via RS-232.
XTPRO 213_50 connected to 3Com USRobotics 56K FaxModem via RS232 (added to standard “&N6” string).

Modem enabled on XTPRO (Par. PAR_BOO_BIOS_18), PAR_ANA_BIOS_193=4, PAR_ANA_BIOS_194=9600, PAR_ANA_BIOS_195=0, PAR_BOO_BIOS_19=1, PAR_BOO_BIOS_20=No,






Modbus Protocol Selections
Transmission Mode
STANDARD
ASCII
RTU
DANIEL/ENRON/OMNI
ASCII
RTU
Slave Response Timeout
1000
(msecs)
Delay Between Polls
20
(msecs)
Force modbus command 15 and 16 for single-point writes.
(To be used in cases where the slave does not support the single-point write functions 05 and 06.)
OK
Cancel

7 PERMITTED AND UNPERMITTED USE

7.1 Permitted Use

For safety purposes, it is important to make sure that the control device is installed and used in accordance with the instructions supplied and that no parts subject to dangerous voltage are accessible to users during ordinary operation.
The unit must be resistant to water and dust, depending on the application, and only be accessible using special tools. This unit can be fitted on domestic appliances and/or similar units used for air conditioning.
In accordance with the reference standards, this unit is classified:

- as an automatic electronic control device to be installed in a standalone configuration or on other units with regard to manufacturing;
- As a Type 1 control unit in relation to its manufacturing tolerances and derivatives with regard to its automatic operating characteristics;
- As a Class 2 device with regard to protection from electric shocks;
- As a Class A device with regard to software class and structure

7.2 Unpermitted Use

The *use* of the unit for applications other than those described is forbidden.

8 DISCLAIMER

8.1 Limited liability

This document is exclusive property of **Eliwell Controls srl** and cannot be reproduced and circulated unless expressly authorized by **Eliwell Controls srl.**
Although **Eliwell Controls srl** has taken all possible measures to guarantee the accuracy of this document, it declines any responsibility for any damage arising out of its *use*.

8.2 AppMaker and WORKBENCH

AppMaker and WORKBENCH are based on IsaGraf software, an ICS Triplex registered trademark. All rights reserved.

9 APPENDICE – PROBLEMS RESOLUTION

9.1 Troubleshooting

9.1.1 No Modbus communication

If *Modbus* communication has failed with Energy XT PRO, check the settings required to restore communication.
A list of instructions is provided below outlining how best to check for potential errors in instrument settings.

9.1.1.1 Setting COM1 configuration parameters

RS-485 *COM1* serial port operation depends on the status of the 3 dedicated parameters PAR_ANA_BIOS_190, PAR_ANA_BIOS_191 and PAR_ANA_BIOS_193**:**

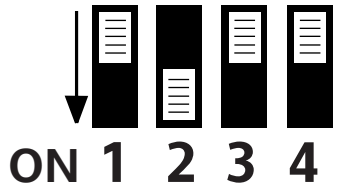
Name of parameter	*Modbus* address [DEC]	Description
PAR_ANA_BIOS_190	222	*COM3* protocol selection 2=*Micronet* 3=*Modbus*/RTU
PAR_ANA_BIOS_191	223	*COM3* baud selection 0=9600 b/s 1=19200 b/s 2=38400 b/s
PAR_ANA_BIOS_192	224	*COM1* parity selection 0=null 1=odd 2=even

Make sure that the first 2 parameters are set as indicated below:

PAR_ANA_BIOS_190**= 3 (*Modbus*/RTU)**
PAR_ANA_BIOS_191 **= 0 (9600 b/s)**

IMPORTANT! The *COM1* parity parameter PAR_ANA_BIOS_192 must be consistent with the parity set in the software to be adopted.

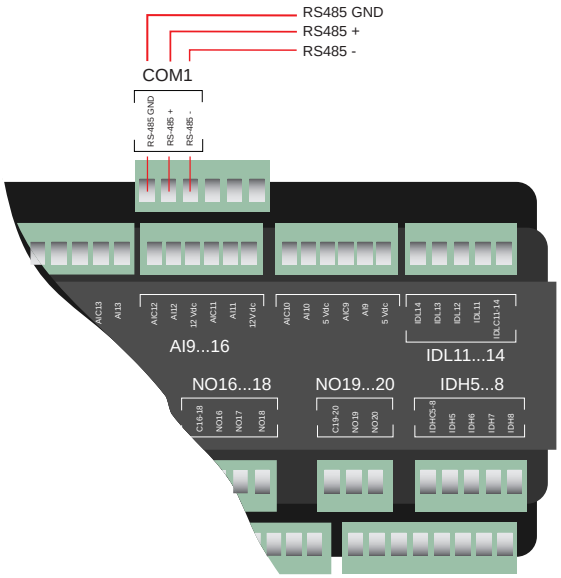
9.1.1.2 Hardware address



The software and *hardware address*es must be the same. To ensure this, make sure DIP SWITCHES are set as follows:
IMPORTANT: To set the *hardware address* to "1" (ON), only the second dip switch from the left needs to be at the bottom.

9.1.1.3 Polarity and Position of port COM1

Check the correct position of serial port *COM1*



9.1.1.4 Passwords

The password must be communicated to Energy XT from a PC to enable the user for communication.
An error entering the password may block communication.
There are three different *types of password* in Energy XT PRO:

- READ PASSWORD enabling read commands only (when “blank” read is always enabled)
- USER PASSWORD enabling read and write commands.
- ADMINISTRATOR PASSWORD enabling read and write commands (not modifiable by USER).

N:B.: on entering the ADMINISTRATOR password, all 3 *passwords* can be changed. Entering the USER password will only allow the USER and READ *passwords* to be changed.

All *passwords* have 10 characters. The default password set by Eliwell has 10 “empty spaces”, meaning that Energy XT serial communication can be enabled by entering a *modbus* write command (command 16) comprising a string of 10 “empty spaces”.

IMPORTANT! Entering a password will enable communication immediately; to disable communication, the password must be re-entered over the serial port.

As described above, serial communication is enabled by entering a string parameter, i.e. the password, but this **enabling of communication is saved in RAM and not EEPROM.** This means that, for example, if power is cut off to Energy XT PRO after the password has been entered and serial communication enabled, when power is returned serial communication will once again be disabled. Hence the password will have to be re-entered to restore serial communication.

9.1.1.5 Area 5 communication test

If you are unable to establish if your PC is actually communicating with Energy XT, we advise you test a particular address area (area 5, STATE area) which, being unprotected by a password, can be read via *modbus* command.
So enter a read command (*modbus* command 3) for the STATE area (area 5).

The structure of the STATE area (area 5) and relevant *modbus* addresses that you'll need to run a read command used for a “communication test" (see the section on Read/Write resources using Modus command 3 and 16,sub-section Read/Write states) are described below:

NP	*Modbus* Address [HEX]	*Modbus* Address [DEC]	Description of element	Value	Always readable	Readable/ Writable only after password recognition
1	8501	34049	Info if parameters have been modified	0: not modified (READ) 0: reset flag (WRITE) 1: modified (READ)	X	X
2	8502	34050	Info presence of active alarms	0: not present 1: present	X	X
3	8503	34051	Not used			
4	8504	34052	Not used			
5	8505	34053	Not used			
6	8506	34054	Update of output state blocked by regulators and input states by drivers.	0: Unlock outputs, always carried out + reset Lock Timeout 1: Lock outputs, carried out only if Lock Timeout is different from 0	X	X
7	8507	34055	Lock Timeout	Time in seconds (max. 600 sec) NOTE: when equal to 0, any Lock is not reset	X	X
8	8508	34056	Not used			
9	8509	34057	Enabling configuration from serial. Used to write COLD parameters	0:_NON_PUOI_RICHIEDERE_INGRESSO_IN_CONFIGURAZIONE_ (READ) (ENTRY TO CONFIGURATION NOT PERMITTED (READ) 1:_NOT USED 2:_ATTENDI_PER_POTER_RICHIEDERE_INGRESSO_IN_CONFIGURAZIONE_ (READ) (WAIT TO OPEN CONFIGURATION) 3:_PUOI_RICHIEDERE_INGRESSO_IN_CONFIGURAZIONE_ (READ) (YOU CAN OPEN CONFIGURATION NOW) 4:_RICHIEDO_CONFIGURAZIONE_ (WRITE) (OPEN CONFIGURATION) 5:_SEI_IN_CONFIGURAZIONE_ (READ) (CONFIGURATION OPEN) 6:_ESCI_DALLA_CONFIGURAZIONE_ (WRITE) (EXIT CONFIGURATION)	X	X

10 APPENDIX-MODBUS MASTER

10.1 Appendix – MODBUS MASTER

Energy XT-PRO is an application that uses the *COM1* and/or *COM3* ports with *Modbus*/RTU protocol on a local network and that is configured as specified in section **Configuration of parameters**.

10.1.1 MODBUS MASTER

If used with the *COM1* (RS485) and *COM3* (RS232) ports, Energy XTPRO can act as master device of a *Modbus*/RTU local network, which means that it is not possible to *use* another modem or the *Modbus*/ASCII protocol.
The combined *use* of *Modbus Master* with the C-Functions implemented in *WORKBENCH* enables to transmit and verify the result of an operation using the error codes. The *Modbus Master* function will therefore be managed through *WORKBENCH* only.
Consequently, when you switch Energy XTPRO on, the *COM1* serial port will made available for the first 15 seconds of start-up for communications with the *WORKBENCH*, while the *COM3* will reply to commands, as per configuration parameters, using an active password protection. After the startup time has expired and the *WORKBENCH* application has been launched, the serial ports will be able to generate *Modbus*/RTU frames upon request of the application, using the available C-Functions.

10.1.2 Parameter configuration

To be able to *use* the *Modbus Master*, you need to correctly set the following parameters:

- When using the *COM1*:

Parameter Name	Description
PAR_ANA_BIOS_187	Family of the device serial address (*)
PAR_ANA_BIOS_190	*COM1* protocol selection 3=Mosbus/RTU
PAR_ANA_BIOS_191	*COM1* baud selection 0=9600b/s (**)
PAR_ANA_BIOS_192	*COM1* parity selection 0=null 1=odd 2=even

If (PAR_ANA_BIOS_192 = 0) then *COM1* STOP BIT = 1

- When using the *COM3*:

Parameter Name	Description
PAR_ANA_BIOS_187	Family of the device serial address (*)
PAR_ANA_BIOS_193	*COM3* protocol selection 3=Mosbus/RTU
PAR_ANA_BIOS_194	*COM3* baud selection 0=9600b/s (**)
PAR_ANA_BIOS_195	*COM3* parity selection 0=null 1=odd 2=even
PAR_BOO_BIOS_19	Selection 7/8 data bits *COM3* 1=8 data bits

If (PAR_ANA_BIOS_195 = 0) then *COM3* STOP BIT = 1

(*):
The board address is the same for serial *COM1* and serial *COM3*.
It is a byte consisting of 2 parts:

- The MSB nibble is the device family and is a parameter saved in the EEPROM (MB_FAA_ADDRESS).
- The LSB nibble is the address that is read by the three dip-switches DIP2,3,4 (it is possible to connect a max. of 8 devices).
  For example: if J2=ON,J3=OFF,J4=OFF, the LSB nibble will be 1.
  For example if J2=ON,J3=OFF,J4=OFF the LSB nibble will be 3

(**):
The value of the parameter has no meaning because the baud rate in this mode is automatically set to increase to 9600b/s.

10.1.3 C-Function for communications

Communications are established by using the following *WORKBENCH* C-Functions:

- mbnetdef
- mb43req
- mb03req
- mb04req
- mb16req
- mbwrres
- mblnbusy
- mbexit
- mbslave

The above-described functions enable to implement the 03, 04,16 and 43 *Modbus* commands and to manage the communication diagnostics.

The *WORKBENCH* C-Functions are identified in two sections:

1. Brief description
2. Technical notes*
3. Parameters*

*2-3 highlighted in the text with Courier New

10.1.4 mbnetdef: modbus master network definition

This function enables the *Modbus Master* driver on the specified serial port (COM), so that the user can declare in the *WORKBENCH* library the integer variables that contain the status of the serial communication established with the slave devices connected to the above-mentioned serial port. Slaves have a consecutive address, starting from the address specified in the function parameter (SlvID). Values of communication statuses are automatically updated in the *WORKBENCH* library by the BIOS.

Technical notes:

```
Name:          - mbnetdef

description:   - This function activates the modbus master driver on the
                 specified serial port (COM)and enables the user to declare in the library of
                 WORKBENCH the integer variables that contain
                 the communication status of
                 serial communication with the slave devices connected to the specified serial port. Slave
                 devices are assigned a progressive address starting from the address set as parameter
                 of the function (SlvID). The values of the communication statuses are automatically
                 are automatically updated by the BIOS in the WORKBENCH library.
                 Communication status variables must be declared by the user
                 as described below:
 ------------
|Integer/Real values|
|-------------------------------------------------------------------------------------------------------------------------
|Name   |Attribute       |Address |Comment
|-------------------------------------------------------------------------------------------------------------------------
|StatusSlave<SlvID>        |[internal, integer]|<IsaAd+0>      |Communication status of the slave with ID = SlvID
|StatusSlave<SlvID+1>      |[internal, integer]|<IsaAd+1>      |Communication status of the slave with ID = SlvID+1
|:
|StatusSlave<SlvID+SlvN-1> |[internal,integer]|<IsaAd+SlvN-1> |Communication status of the slave with ID = SlvID+SlvN-1
 -------------------------------------------------------------------------------------------------------------------------

                 Each communication status variable can acquire the following values:

                 0  = Slave not queried in reading or writing mode
                 1  = Slave that has replied to a reading request, but has not yet been used for writing
                 2  = Slave that has returned an exception to the last reading request, but has not yet been used for writing
                 3  = Slave in timeout reading, not yet used for writing
                 16 = Slave that has correctly replied to the last writing request, but not yet queried in reading mode
                 17 = Slave that has correctly replied to the last writing and reading request
                 18 = Slave that has correctly replied to the last writing and reading
                      request
                 19 = Slave that has replied correctly to the last writing request and that was in reading timeout
```

32 = Slave that has returned an exception to the last writing request and has not yet been queried in reading mode
33 = Slave that has returned an exception to the last writing and reading request
34 = Slave that has returned an exception to the last writing and reading request
35 = Slave that has returned an exception to the last writing request and that was in reading timeout
48 = Slave that was in timeout during last writing request and not yet been queried
49 = Slave that was in timeout during last writing request and has replied correctly to last reading request
50 = Slave that was in timeout during last writing request and that has replied to the last reading request
51 = Slave that was in timeout during last reading and reading query

Creation date: - 20/01/05

Author: - Eliwell

Call: - COM : 1 = *COM1* or 3=*COM3*
IsaAd : *Modbus* address of the first integer variable of *WORKBENCH* that will be used as starting point to copy the communication statuses of the slaves connected to the specified COM [16#0193...16#19C7]
SlvN : Number of slave connections to the specified COM. This number must correspond to the difference between the larger serial address and the smaller slave belonging to the network, plus one. The potential range is [1...255]
SlvID : The smaller of all the serial addresses (ID) of the slaves connected to the specified COM. The range is [0...255]

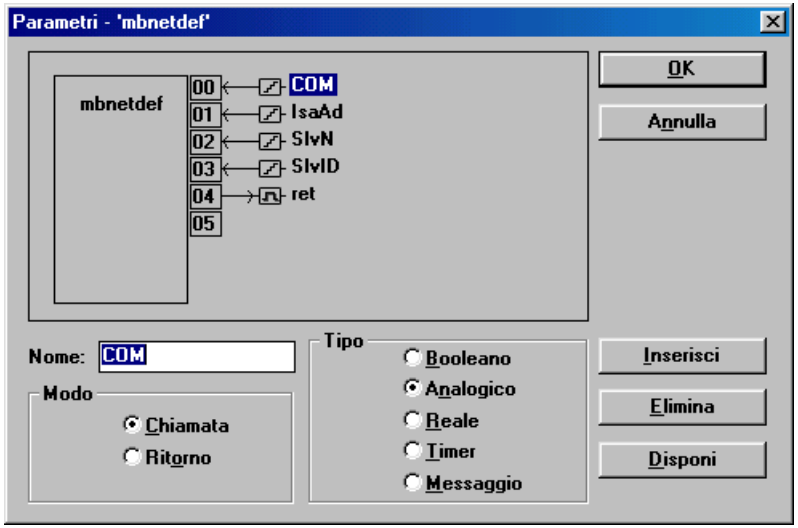
Return: - ret : If TRUE, the command is accepted; otherwise the command is not accepted if the parameters are out of range or if the configuration parameters of the specified serial port (COM) are not compatibile with the *modbus master* is active.

Prototype: - ret := mbnetdef(COM,IsaAd,SlvN,SlvID);

notes: - Status communication codes can be interpreted as follows:
Bits from 0 to 3 of status variables reflect the status of the last query in reading mode. Bits from 4 to 7 reflect the status of the last query in writing mode. The 4 bits of the two groups contain the codes from 0 to 3, which have the following meaning:
0 = Command not sent
1 = Successful command
2 = Exception returned to the command
3 = No reply to the command (timeout)
Therefore, value 19 can be interpreted as 16#0013, which means that the last writing operation was successful 1) and that a timeout has occurred during the last query

```
                in reading mode (3).

Example:       - See below:

(* COM1 is connected to two slaves devices that have serial addresses 3 and 4. Their communication status *)
(* will be specified in the library of integer values of WORKBENCH starting from the integer           *)
(* with modbus address 16#200.                                                                        *)
ret := mbnetdef(1,16#200,2,3);


This function works correctly only if the user has declared in the library of integers of
of WORKBENCH the following integer variables:

 ------------
|Integer/Real values|
|-------------------------------------------------------------------------------------------------
|Name          |Attribute       |Address|Comment
|-------------------------------------------------------------------------------------------------
|StatusSlave3  |[internal,integer]|0200     |Communication status of slave with ID = 3
|StatusSlave4  |[internal,integer]|0201     |Communication status of slave with ID = 4
 -------------------------------------------------------------------------------------------------
```

Parameters



10.1.5 mb43req: modbus master command 43 request

This function enables the Master device to read the ID and the additional information of one or more slaves with contiguous addresses (ID), using the “individual access” mode of *command 43* (Read Device Identification request). For the read strings to be accessible to the *WORKBENCH* application, users have to declare the messages that contain the value of returned ASCII strings in the *WORKBENCH* library. These messages are automatically updated by the BIOS. The start *Modbus* address of these messages is specified in the function call (IsaAd), along with the ID code of the information that has to be read (ObjID).

Technical notes:

Name: - **mb43req**

description: - **This function enables the master device to read the IDs and additional information of one or more slaves with contiguous serial addresses (ID) in accordance with the "individual access" mode of *command 43* (Read Device Identification request). To make the read strings available to *WORKBENCH*, the user must declare in the *WORKBENCH* library the messages that contain the ASCII strings. These messages are automatically updated by the BIOS. The starting *modbus* address of these messages is specified in the call to the function (IsaAd), along with the ID code of the information to read (ObjID). The section that follows explains how to define variables for the messages described above.**

Messages

Name	Attribute	Address	Comment
infoSlave<SlvID>	[internal]	<IsaAd+0>	ASCII string of the information with code ObjID of slave with ID = SlvID
infoSlave<SlvID+1>	[internal]	<IsaAd+1>	ASCII string of the information with code ObjID of slave with ID = SlvID+1
:			
infoSlave<SlvID+SlvN-1>	[internal]	<IsaAd+SlvN-1>	ASCII string of the information with code ObjID of slave with ID = SlvID+SlvN-1

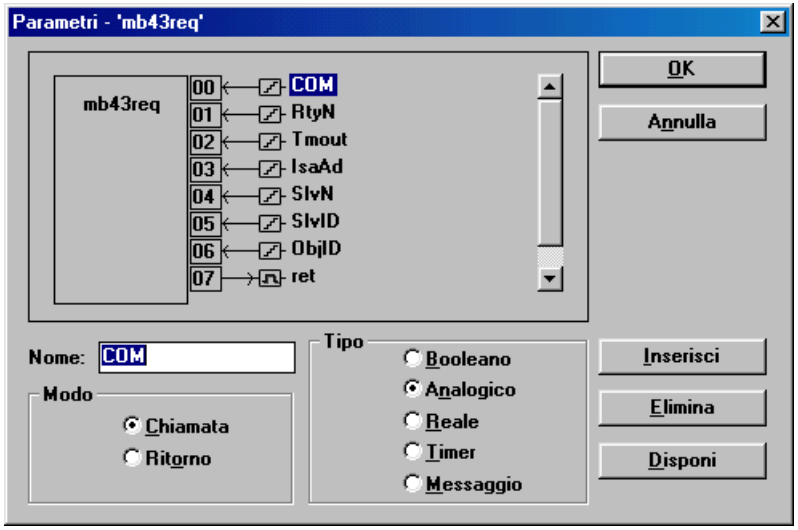
Creation date: - **07/01/05**

Author: - **Eliwell**

Call: - **COM : 1 = *COM1* or 3 = *COM3***
RtyN : Number of retries after the timeout of the first request [0...255]
Tmout : Timeout between request and reply [0...2000ms]
IsaAd : *Modbus* address in *WORKBENCH*, used to copy the read strings [16#0193...16#19C7]
SlvN : Number of slaves to query. This number must be greater than zero. The upper limit changes according to the available memory. The potential range is [1...255]

```
                SlvID : Serial ID (ID) of the first slave to query. The first slave is the one with the lowest
                        ID. The list of slave IDs starts from ID=SlvID
                        and ends with ID=SlvID+SlvN. The range is [0...255]
                ObjID   : ID code of the information [16#00...16#FF]

Return:       - ret   : If TRUE, the function is always run; if FALSE the function is not run because
                      the parameters are out of range or because the configuration parameters of the specified serial port
                      (COM) are not compatible with the modbus master mode or because the modbus master
                      driver has not been enabled on the port or because the serial port is busy
                      is busy with modbus queries.

Prototype:    - ret := mb43req(COM,RtyN,Tmout,IsaAd,SlvN,SlvID,ObjID);

notes:        - Messages must be declared in the dictionary of WORKBENCH so that their length
                is greater or equal to the requested ASCII string.

Example:      - See below:

(*COM1 is used to read from slaves with serial addresses 3 and 4    *)
(* the objects that identify MajorMinorRevision (object code = 2).        *)
(* Their values will be copied in the library of messages of WORKBENCH starting *)
(* from the message with Modbus address 16#202. The number of acceptable attempts after the first one  *)
(* before declaring the disconnection of the queried salve is 3, while the timeout *)
(* between a request and a reply is 200ms.                    *)
ret := mb43req(1,3,200,16#202,2,3,2);


This function works correctly only if the user has declared in the library of integers
of WORKBENCH the following message variables:

 --------
|Messages|
|-------------------------------------------------------------------------------------------------
|Name   |Attribute       |Address |Comment
|-------------------------------------------------------------------------------------------------
|MMRevSlave3  |[internal] |0202      |Major Minor Revision string of slave with ID = SlvID
|MMRevSlave4  |[internal] |0203      |Major Minor Revision string of slave with ID = SlvID+1
 -------------------------------------------------------------------------------------------------
```

Parameters:


Parametri - 'mb43req'
mb43req
00 COM
01 RtyN
02 Tmout
03 IsaAd
04 SlvN
05 SlvID
06 ObjID
07 ret
OK
Annulla
Nome: COM
Modo
Chiamata
Ritorno
Tipo
Booleano
Analogico
Reale
Timer
Messaggio
Inserisci
Elimina
Disponi

10.1.6 mb03req: modbus master command 03 request

This function of the Master device reads the blocks of contiguous registers from one or more slaves with contiguous address. This is done by sending to the specified serial port (COM) *Modbus* packets formatted with command 03 (Read Holding Register). Blocks can be read in two different ways: by reading, once only, the registers of the specified slaves (AutoScan=FALSE) or by continuing to read them endlessly by means of subsequent scans that are automatically managed by the BIOS (AutoScan=TRUE). For the read values to be accessible to the *WORKBENCH* application, users have to declare the messages that contain the register values in the *WORKBENCH* library. These variables will be automatically updated by the BIOS. The start *Modbus* address of these integer variables is specified in the call of the function (IsaAd), along with the number of registers to read (RgN) and the *Modbus* address that must be used as start address (StrAd) on the slave. The communication status variables of the slaves can be used to monitor the reading operation and verify that it has been successfully completed. For further details, see function mbnetdef.

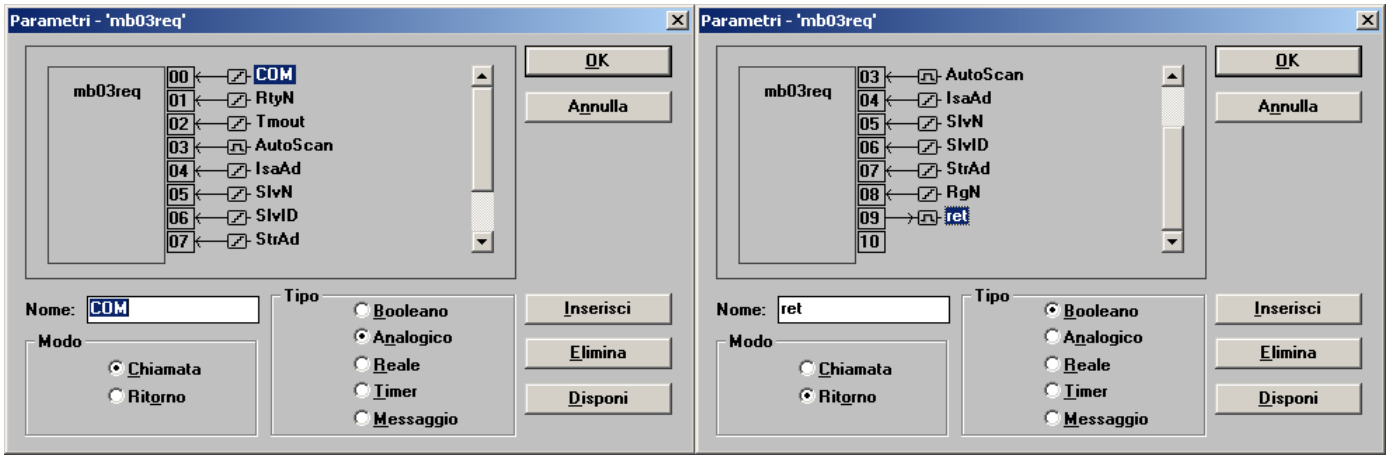
Technical notes:

```
Name:          - mb03req

description:   - This function of the master device reads the blocks of contiguous registers from
                 one or more slaves with a contiguous address. This is done by sending
                 to the specified serial port (COM) modbus packages formatted with
                 command 03 (Read Holding Register).Data can be read in two different
                 ways: by reading the registers of the specified slave devices once only
                 (AutoScan=FALSE) or by continuing to read them through a series of subsequent
                 scans that are managed automatically by the BIOS (AutoScan=TRUE).
                 To make the read values accessible to WORKBENCH, users  must
                 declare in the WORKBENCH library integer variables that contain the
                 values of registers. These variables are automatically updated
                 by the BIOS. The starting modbus address of these integer variables
                 is specified in the call to the function(IsaAd), along with the number
                 of registers to read (RgN) and the modbus address from where you need start to reading them
                 (StrAd) on the slave. The communication status variables with slaves
                 enable to verify if the reading operation has been successful
                 or not. For further details, see function mbnetdef.
                 The section below specifies how to define integer variables that contain
                 the values of read registers.

 ------------
|Integer/Real values|
|-------------------------------------------------------------------------------------------------------------------------------
|Name   |Attribute       |Address |Comment
|-------------------------------------------------------------------------------------------------------------------------------
|Reg<SlvID>1              |[internal,integer]|<IsaAd+1>                        |Value of the register with StrAd modbus address
of slave with ID=SlvID
|Reg<SlvID>2              |[internal,integer]|<IsaAd+2>                        |Value of the register with StrAd+1 modbus address
of slave with ID=SlvID
|:
|Reg<SlvID><RgN-1>        |[internal,integer]|<IsaAd+(RgN-1)>                  |Value of the register with StrAd+RgN-1 modbus
address of slave with ID=SlvID
```

```
|Reg<SlvID+1>1            |[internal,integer]|<IsaAd+(RgN-1)+1>                |Value of the register with StrAd modbus address
of slave with ID=SlvID+1
|Reg<SlvID+1>2            |[internal,integer]|<IsaAd+(RgN-1)+2>                |Value of the register with StrAd+1 modbus address
of slave with ID=SlvID+1
|:
|Reg<SlvID+1><RgN-1>      |[internal,integer]|<IsaAd+(RgN-1)+(RgN-1)>          |Value of the register with StrAd+RgN-1 modbus
address of slave with ID=SlvID+1
|:
|Reg<SlvID+SlvN-1>1       |[internal,integer]|<IsaAd+(RgN-1)*(SlvN-1)>         |Value of the register with StrAd modbus address
of slave with ID=SlvID+SlvN-1
|Reg<SlvID+SlvN-1>2       |[internal,integer]|<IsaAd+(RgN-1)*(SlvN-1)+1>       |Value of the register with StrAd+1 modbus address
of slave with ID=SlvID+SlvN-1
|:
|Reg<SlvID+SlvN-1><RgN-1> |[internal,integer]|<IsaAd+(RgN-1)*(SlvN-1)+(RgN-1)> |Value of the register with StrAd+RgN-1 modbus
address of slave with ID=SlvID+SlvN-1
 ------------------------------------------------------------------------------------------------------------------------

Creation date: - 07/01/05

Author:        - Eliwell

Call:          - COM    : 1 = COM1 or 3 = COM3
                 RtyN     : Number of retries after the timeout of the first request [0...255]
                 Tmout    : Timeout between request and reply [0...2000ms]
                 AutoScan : If TRUE, the registers are read from all the specified slaves and the scan restarts automatically
                            If FALSE, the query of the slaves been scanned is interrupted [TRUE,FALSE]
                 IsaAd    : Modbus address in WORKBENCH, used to copy the values of the read registers [16#0193...16#19C7]
                 SlvN     : Number of slaves to query. This number must be greater than zero. The upper limit changes
                            according to the available memory. The potential range is [1...255]
                 SlvID    : Serial ID (ID) of the first slave to query. The first slave is the one with the lowest
                            ID. The list of slave IDs starts from ID=SlvID
                            and ends with ID=SlvID+SlvN. The range is [1...255]
                 StrAd    : Modbus address of the first register that must be read for the queries slave [16#0000...16#FFFF]
                 RgN      : Number of contiguous registers, starting from the one with address StrAd, that must be read
                            from the queried slave [1...125].

Return:        - ret   : If TRUE, the function is always run; if FALSE the function is not run because
                       the parameters are out of range or because the configuration parameters of the specified serial port
                       (COM) are not compatible with the modbus master mode or because the modbus master
                       driver has not been enabled on the port or because the serial port is busy
                       is busy with modbus queries.
```

```
Prototype:     - ret := mb03req(COM,RtyN,Tmout,AutoScan,IsaAd,SlvN,SlvID,StrAd,RgN);

notes:         - These registers are 16 bit registers with sign. Therefore, if the
                 slave reads a larger variable, the value inserted
                 in the library of WORKBENCH will be its truncated value
                 with sign.

Example:       - See below:

(* COM1 is used to continuously read from slaves with address *)
(* 3 and 4 the two registers 16#8501 and 16#8502.                    *)
(* Their values will be copied in the library of integer values of  WORKBENCH starting from *)
(* from  the integer with Modbus address 16#204. The number of acceptable attempts after the first one  *)
(* before declaring the disconnection of the queried salve is 3, while the timeout *)
(* between a request and a reply is 200ms.                          *)
ret := mb03req(1,3,200,TRUE,16#204,2,3,16#8501,2);


This function works correctly only if the user has declared in the library of integers
of WORKBENCH the following integer variables:

 ------------
|Integer/Real values|
|-------------------------------------------------------------------------------------------------------------
|Name   |Attribute       |Address |Comment
|-------------------------------------------------------------------------------------------------------------
|Reg31  |[internal,integer]|0204      |Value of the register with modbus address 16#8501 of slave with ID=3
|Reg32  |[internal,integer]|0205      |Value of the register with modbus address 16#8502 of slave with ID=3
|Reg41  |[internal,integer]|0206      |Value of the register with modbus address 16#8501 of slave with ID=4
|Reg42  |[internal,integer]|0207      |Value of the register with modbus address 16#8502 of slave with ID=4
 -------------------------------------------------------------------------------------------------------------
```

Parameters:


Parametri - 'mb03req'
mb03req
00 COM
01 RtyN
02 Tmout
03 AutoScan
04 IsaAd
05 SlvN
06 SlvID
07 StrAd
OK
Annulla
Nome: COM
Modo
Chiamata
Ritorno
Tipo
Booleano
Analogico
Reale
Timer
Messaggio
Inserisci
Elimina
Disponi
Parametri - 'mb03req'
mb03req
03 AutoScan
04 IsaAd
05 SlvN
06 SlvID
07 StrAd
08 RgN
09 ret
10
OK
Annulla
Nome: ret
Modo
Chiamata
Ritorno
Tipo
Booleano
Analogico
Reale
Timer
Messaggio
Inserisci
Elimina
Disponi

10.1.7 mb04req: modbus master command 04 request

This function of the Master device reads the blocks of contiguous input registers from one or more slaves with contiguous address. This is done by sending to the specified serial port (COM) *Modbus* packets formatted with command 04 (Read Input Register). Blocks can be read in two different ways: by reading, once only, the registers of the specified slaves (AutoScan=FALSE) or by continuing to read them endlessly by means of subsequent scans that are automatically managed by the BIOS (AutoScan=TRUE). For the read values to be accessible to the *WORKBENCH* application, users have to declare the messages that contain the register values in the *WORKBENCH* library. These variables will be automatically updated by the BIOS. The start *Modbus* address of these integer variables is specified in the call of the function (IsaAd), along with the number of registers to read (RgN) and the *Modbus* address that must be used as start address (StrAd) on the slave. The communication status variables of the slaves can be used to monitor the reading operation and verify that it has been successfully completed. For further details, see function mbnetdef.

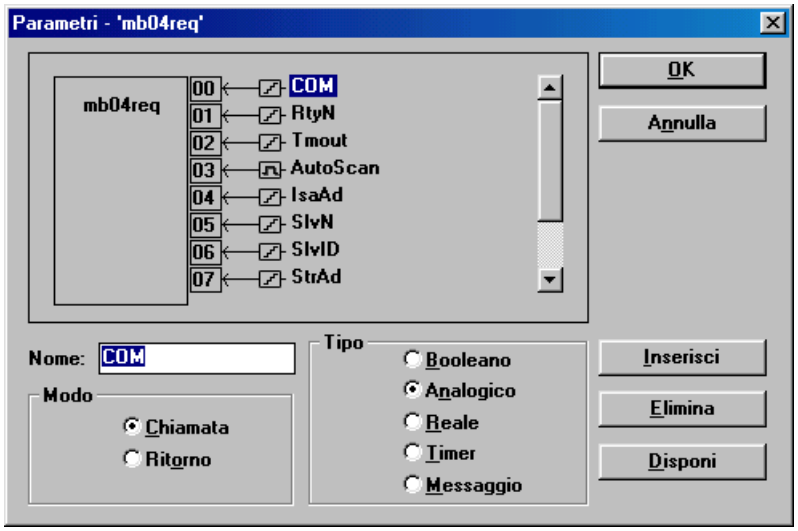
Technical notes:

```
Name:          - mb04req

description:   - This function of the master device reads the contiguous input blocks from
                 one or more slaves with contiguous address. This is done by sending
                 to the specified serial port (COM) modbus packages formatted with
                 command 04 (Read Input Register).Data can be read in two different
                 ways: by reading the registers of the specified slave devices once only
                 (AutoScan=FALSE) or by continuing to read them through a series of subsequent
                 scans that are managed automatically by the BIOS (AutoScan=TRUE).
                 To make the read values accessible to WORKBENCH, users  must
                 declare in the WORKBENCH library integer variables that contain the
                 values of registers. These variables are automatically updated
                 by the BIOS. The starting modbus address of these integer variables
                 is specified in the call to the function(IsaAd), along with the number
                 of registers to read (RgN) and the modbus address from where you need start reading them
                 (StrAd) on the slave. The communication status variables with slaves
                 enable to verify if the reading operation has been successful
                 or not. For further details, see function mbnetdef.
                 The section below specifies how to define integer variables that contain
                 the values of read registers.

 ------------
|Integer/Real values|
|-------------------------------------------------------------------------------------------------------------------------------
|Name   |Attribute       |Address |Comment
|-------------------------------------------------------------------------------------------------------------------------------
|Reg<SlvID>1             |[internal,integer]|<IsaAd+1>                      |Value of the register with StrAd modbus address
of slave with ID=SlvID
|Reg<SlvID>2             |[internal,integer]|<IsaAd+2>                      |Value of the register with StrAd+1 modbus address
of slave with ID=SlvID
|:
|Reg<SlvID><RgN-1>       |[internal,integer]|<IsaAd+(RgN-1)>                |Value of the register with StrAd+RgN-1 modbus
address of slave with ID=SlvID
```

```
|Reg<SlvID+1>1            |[internal,integer]|<IsaAd+(RgN-1)+1>                |Value of the register with StrAd modbus address
of slave with ID=SlvID+1
|Reg<SlvID+1>2            |[internal,integer]|<IsaAd+(RgN-1)+2>                |Value of the register with StrAd+1 modbus address
of slave with ID=SlvID+1
|:
|Reg<SlvID+1><RgN-1>      |[internal,integer]|<IsaAd+(RgN-1)+(RgN-1)>          |Value of the register with StrAd+RgN-1 modbus
address of slave with ID=SlvID+1
|:
|Reg<SlvID+SlvN-1>1       |[internal,integer]|<IsaAd+(RgN-1)*(SlvN-1)>         |Value of the register with StrAd modbus address
of slave with ID=SlvID+SlvN-1
|Reg<SlvID+SlvN-1>2       |[internal,integer]|<IsaAd+(RgN-1)*(SlvN-1)+1>       |Value of the register with StrAd+1 modbus address
of slave with ID=SlvID+SlvN-1
|:
|Reg<SlvID+SlvN-1><RgN-1> |[internal,integer]|<IsaAd+(RgN-1)*(SlvN-1)+(RgN-1)> |Value of the register with StrAd+RgN-1 modbus
address of slave with ID=SlvID+SlvN-1
 ------------------------------------------------------------------------------------------------------------------------

Creation date: - 07/01/05

Author:        - Eliwell

Call:          - COM    : 1 = COM1 or 3 = COM3
                 RtyN   : Number of retries after the timeout of the first request [0...255]
                 Tmout  : Timeout between request and reply [0...2000ms]
                 AutoScan : If TRUE, the registers are read from all the specified slaves and the scan restarts automatically
                            If FALSE, the query of the slaves been scanned is interrupted [TRUE,FALSE]
                 IsaAd  : Modbus address in WORKBENCH, used to copy the values of the read registers [16#0193...16#19C7]
                 SlvN   : Number of slaves to query. This number must be greater than zero. The upper limit changes
                            according to the available memory. The potential range is [1...255]
                 SlvID : Serial ID (ID) of the first slave to query. The first slave is the one with the lowest
                            ID. The list of slave IDs starts from ID=SlvID
                            and ends with ID=SlvID+SlvN. The range is [1...255]
                 StrAd  : Modbus address of the first register that must be read for the queries slave [16#0000...16#FFFF]
                 RgN    : Number of contiguous registers, starting from the one with address StrAd, that must be read
                            from the queried slave [1...125].

Return:        - ret   : If TRUE, the function is always run; if FALSE the function is not run because
                       the parameters are out of range or because the configuration parameters of the specified serial port
                       (COM) are not compatible with the modbus master mode or because the modbus master
                       driver has not been enabled on the port or because the serial port is busy
                       is busy with modbus queries.
```

```
Prototype:     - ret := mb04req(COM,RtyN,Tmout,AutoScan,IsaAd,SlvN,SlvID,StrAd,RgN);

notes:         - These registers are 16 bit registers with sign. Therefore, if the
                 slave reads a larger variabale, the value inserted
                 in the library of WORKBENCH will be its truncated value
                 with sign.

Example:       - See below:

(* COM1 is used to continuously read from slaves with address *)
(* 3 and 4 the two registers 16#8501 and 16#8502.                    *)
(* Their values will be copied in the library of integer values of  WORKBENCH starting from *)
(* from  the integer with Modbus address 16#204. The number of acceptable attempts after the first one  *)
(* before declaring the disconnection of the queried salve is 3, while the timeout *)
(* between a request and a reply is 200ms.                          *)
ret := mb04req(1,3,200,TRUE,16#204,2,3,16#8501,2);


This function works correctly only if the user has declared in the library of integers
of WORKBENCH the following integer variables:

 ------------
|Integer/Real values|
|-------------------------------------------------------------------------------------------------------------
|Name   |Attribute       |Address |Comment
|-------------------------------------------------------------------------------------------------------------
|Reg31  |[internal,integer]|0204      |Value of the register with modbus address 16#8501 of slave with ID=3
|Reg32  |[internal,integer]|0205      |Value of the register with modbus address 16#8502 of slave with ID=3
|Reg41  |[internal,integer]|0206      |Value of the register with modbus address 16#8501 of slave with ID=4
|Reg42  |[internal,integer]|0207      |Value of the register with modbus address 16#8502 of slave with ID=4
 -------------------------------------------------------------------------------------------------------------
```

Parameters:


Parametri - 'mb04req'
mb04req
00 COM
01 RtyN
02 Tmout
03 AutoScan
04 IsaAd
05 SlvN
06 SlvID
07 StrAd
OK
Annulla
Nome: COM
Modo
Chiamata
Ritorno
Tipo
Booleano
Analogico
Reale
Timer
Messaggio
Inserisci
Elimina
Disponi

10.1.8 mb16req: modbus master command 16 request

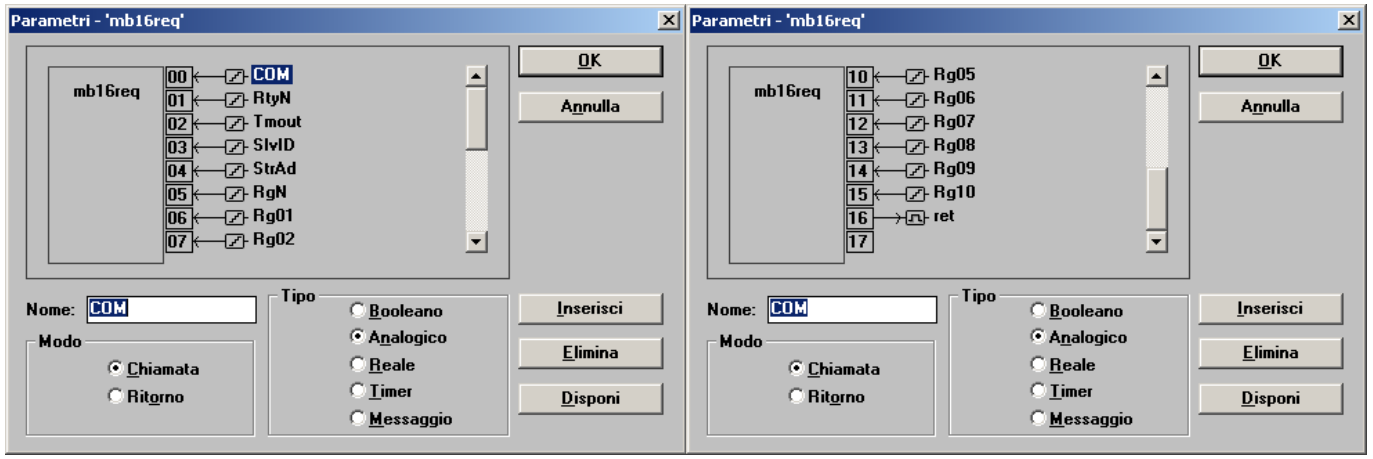
This function is used by the Master device to write a contiguous block of registers of a slave with serial ID (SlvID), connected to the specified serial port (COM). It is possible to *use* up to a maximum of 10 registers (maximum value of RgN). Calling the function is equivalent to sending a *Modbus* package formatted with command 16 (Write Multiple registers request). This transmission has priority over data reading operations that are in progress on the specified serial port, but not over writing operations. The result of the writing operation can be controlled with the mbwrres function and the communication status variables of the slaves. For further details, see functions mbwrres and mbnetdef.

Technical notes:

```
Name:          - mb16req

description:   - This function is used by the master device to write a contiguous block of registers
                 of a slave with ID serial address (SlvID) connected to the specified serial
                 device (COM. It is possible to use up to a maximum of 10 registers (maximum value of RgN).
                 The call of this function is similar to sending a modbus package formatted
                 with command 16 (Write Multiple registers request).
                 This transmission has priority over other data reading operations
                 in progress on the specified serial port, but not over other writing operations.
                 The result of the writing operation can be controlled by means of function
                 mbwrres and by means of the communication status variable with the slave devices.
                 For further details, see functions mbwrres and mbnetdef.

Creation date: - 07/01/05

Author:        - Eliwell

Call:          - COM    : 1 = COM1 or 3 = COM3
                 RtyN     : Number of retries after the timeout of the first request [0...255]
                 Tmout  : Timeout between request and reply [0...2000ms]
                 SlvID : Serial address (ID) of the slave selected for the writing operation [0...255]
                 StrAd    : Modbus address of the first register to be used to write to the specified slave [16#0000...16#FFFF]
                 RgN      : Number of registers to write [1...10]
                 Rg01     : Value of the first register to write [-32768...+32767]
                 Rg02     : Value of the second register to write [-32768...+32767]
                 Rg03     : Value of the third register to write [-32768...+32767]
                 Rg04     : Value of the fourth register to write [-32768...+32767]
                 Rg05     : Value of the fifth register to write [-32768...+32767]
                 Rg06     : Value of the sixth register to write[-32768...+32767]
                 Rg07     : Value of the seventh register to write [-32768...+32767]
                 Rg08     : Value of the eighth register to write [-32768...+32767]
                 Rg09     : Value of the ninth register to write [-32768...+32767]
                 Rg10     : Value of the tenth register to write [-32768...+32767]
```

```
Return:        - ret   : If TRUE, the function is always run; if FALSE the function is not run because
                         the parameters are out of range or because the configuration parameters of the specified serial port
                         (COM) are not compatible with the modbus master mode or because the modbus master
                         driver has not been enabled on the port or because the serial port is busy
                         with other writing requests..

Prototype:     - ret := mb16req(COM,RtyN,Tmout,SlvID,StrAd,RgN,Rg01,Rg02,Rg03,Rg04,Rg05,Rg06,Rg07,Rg08,Rg09,Rg10);

notes:         - Before calling function mb16req, it is advisable to call function mbwrres to
                 verify that there are no writing operations in progress.
                 The registers specified above are 16-bit registers with sign.

Example:       - See below:

(* COM1 is used to write on slave with serial address  *)
(* 4 due two registers 16#018C and 16#018D with values *)
(* 0 and 30. The number of accepted re-attempts after the first before declaring     *)
(* the disconnection of the queried slave is 3 and the timeout      *)
(* between a request and a reply is 200ms.     *)
ret := mb16req(1,3,200,4,16#018C,2,0,30,0,0,0,0,0,0,0,0);
```

Parameters:



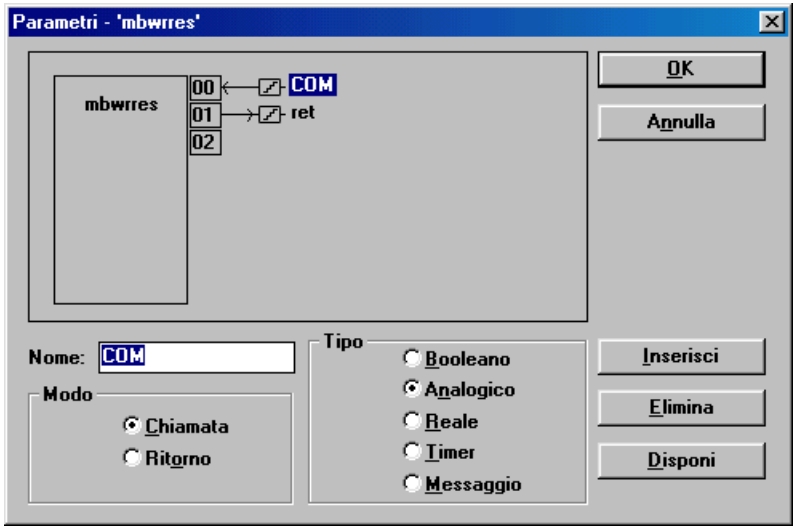
10.1.9 mbwrres: modbus master write response

This function is used to test the reply to a previous writing request sent to the specified Master serial port (COM).

Technical notes:

```
Name:          - mbwrres

description:   - This function is used to test the reply to a previous
                 writing request sent to the specified master serial
                 device (COM.

Creation date: - 20/01/05

Author:        - Eliwell

Call:          - COM    : 1 = COM1 or 3 = COM3

Return:        - ret   : 0 = No writing command sent; it is possible to send a writing command
                         1 = The previous writing command has been successful; it is possible to send another writing command
                         2 = Exception returned to previous writing command; it is possible to send another writing command
                         3 = Timeout for previous command; it is possible to send another writing command
                         4 = Writing command in progress; it is not possible to send another writing command
                         8 = Error: the function cannot be run because the parameters are out of range or because the configuration
                             parameters of the specified serial device (COM) are not compatible with the modbus master mode or
because
                             the modbus master driver has not been enabled on the serial port.

Prototype:     - ret := mbwrres(COM);

notes:         - None.

Example:       - None.
```

Parameters:


Parametri - 'mbwrres'
mbwrres
00
01
02
COM
ret
OK
Annulla
Nome:
COM
Modo
Chiamata
Ritorno
Tipo
Booleano
Analogico
Reale
Timer
Messaggio
Inserisci
Elimina
Disponi

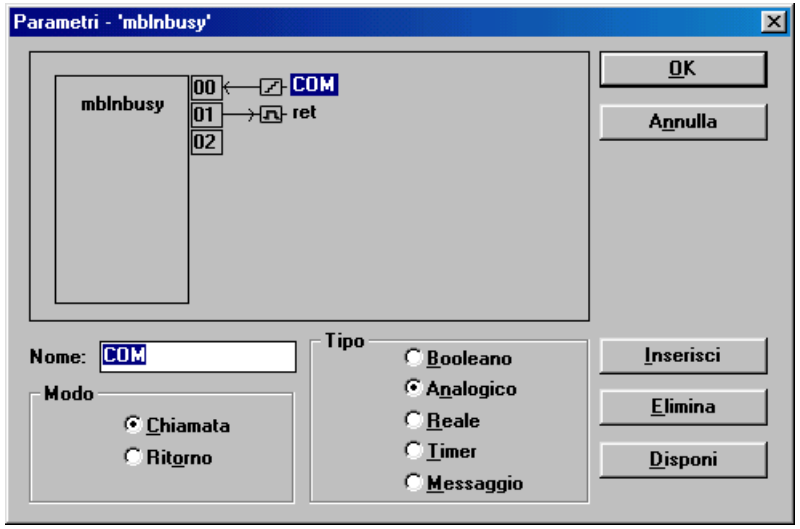
10.1.10 mblnbusy: modbus master line busy

This function is used to verify if the specified serial port (COM) is used for reading or writing operations. See also the mbslave function.

Technical notes:

```
Name:          - mblnbusy

description:   - This function is used to check if the specified serial port (COM) is
                 Busy with a reading or writing operation.

Creation date: - 07/01/05

Author:        - Eliwell

Call:          - COM    : 1 = COM1 or 3=COM3

Return:        - ret   : If TRUE, the line is busy, i.e. there is traffic
                       If FALSE, the line is free or the function cannot be run because
                       the parameters are out range

Prototype:     - ret := mblnbusy(COM);

notes:         - If the WORKBENCH application has enabled functions with AutoScan = TRUE,
                 the function always returns a busy line if there is no error. See also
                 mbslave.

Example:       - None.
```

Parameters:


Parametri - 'mblnbusy'
mblnbusy
00
01
02
COM
ret
OK
Annulla
Nome:
COM
Modo
Chiamata
Ritorno
Tipo
Booleano
Analogico
Reale
Timer
Messaggio
Inserisci
Elimina
Disponi

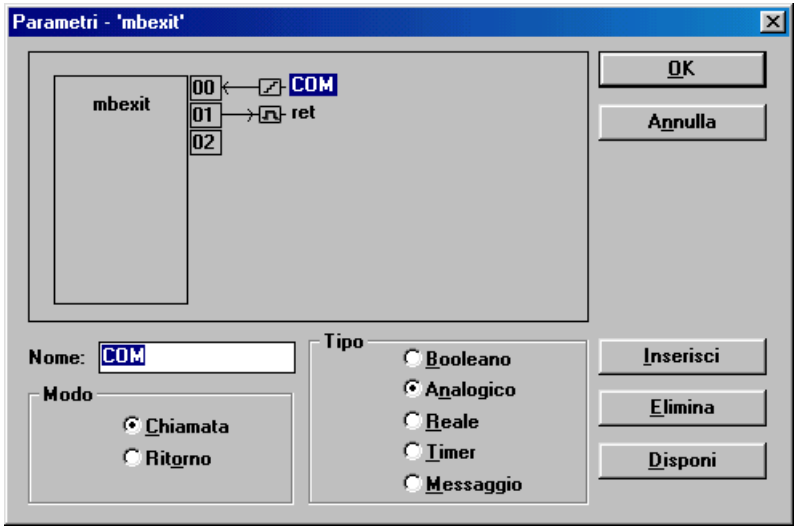
10.1.11 mbexit: : modbus master mode exit

This function is used to allow the specified serial port (COM) to exit from the serial function. All communications in progress are interrupted. Interruption occurs immediately if the communication has been requested, but not yet started. Otherwise (communication in progress), the *Modbus Master* function is disabled as soon as a reply is received or a timeout occurs on the slave on which communication is in progress. This function works in combination with mblnbusy. If this is tested after the call of mbexit, it enables to establish if communications have been interrupted and if the Master mode has been disabled for the specified serial port. To be able to reuse the serial port for master tasks, you first need to recall the mbnetdef function.

Technical notes:

```
Name:          - mbexit

description:   - This function is used to exit from the modbus master mode when using
                 the specified serial port (COM). All communications in progress are interrupted.
                 The interruption occurs immediately if the communication has been requested, but not
                 yet started. Otherwise (communication in progress), the device exits from the
                 modbus master mode as soon as the reply is received or if a timeout of the command
                 related to the slave communicating occurs.
                 This function works in combination with mblnbusy. This function tests
                 the call to mbexit in order to verify if communications have been interrupted and
                 if the specified serial device has exited from the master mode.
                 To be able to reuse the serial device for master activities, you first need
                 to call the mbnetdef function.


Creation date: - 07/01/05

Author:        - Eliwell

Call:          - COM    : 1 = COM1 or 3=COM3

Return:        - ret   : If TRUE, the exit request command has been accepted; otherwise
                       (FALSE) the function cannot be run because the parameters are out of range.

Prototype:     - ret := mbexit(COM)

notes:         - If the specified serial device is not in master mode, the function returns TRUE.

Example:       - None.
```

Parameters:


Parametri - 'mbexit'
mbexit
00
01
02
COM
ret
OK
Annulla
Nome:
COM
Modo
Chiamata
Ritorno
Tipo
Booleano
Analogico
Reale
Timer
Messaggio
Inserisci
Elimina
Disponi

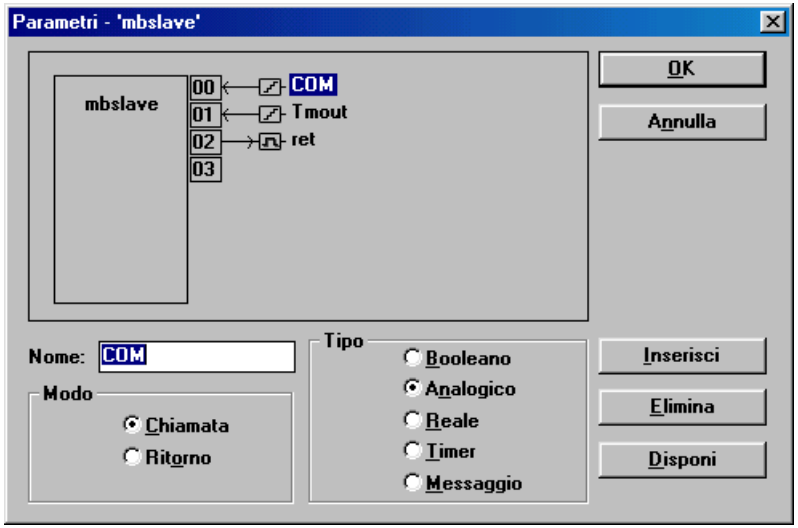
10.1.12 mbslave: modbus master slave mode additional proprieties

This function can be used on a serial port (COM) where no *Modbus Master* mode has been enabled in order to:
Introduce the management of a connection timeout with the corresponding Master (Tmout) and monitor it with the mblnbusy function.
The password management function on the serial port is disabled. In other words, it is always possible to read and write using all the available *Modbus* addresses, except those related to all *passwords*. In this case, the function will return an exception.

Technical notes:

```
Name:          - mbslave

description:   - This function can be used on a serial port (COM) where no
                 modubus master mode is active in order to:
                 1) Introduce the management of the connection timeout with the related
                    master (Tmout), which can be monitored through function mblnbusy.
                 2) The password management on the specified serial port is disabled. This means
                    that is always possible to read and write using all the available modbus addresses,
                    except those used for passwords. In this case, the function will return an exception.

Creation date: - 07/01/05

Author:        - Eliwell

Call:          - COM    : 1 = COM1 or 3=COM3
                 Tmout  : Connection timeout. Indicates after how long the query is sent
                          to the specified serial port [0...60sec].

Return:        - ret    : If TRUE, the command is accepted; otherwise the command is not accepted
                          because the parameters are out of range or the configuration parameters
                          of the specified serial port (COM) are not compatible or because the
                          modbus master is active.

Prototype:     - ret := mbslave(COM,Tmout);

notes:         - If the timeout has expired, the mblnbusy function returns FALSE, otherwise TRUE. To re-enable
                 the modbus master mode on the specified serial port, it is necessary to call
                 mbnetdef.

Example:       - None.
```

Parameters:


Parametri - 'mbslave'
mbslave
00
01
02
03
COM
Tmout
ret
OK
Annulla
Nome:
COM
Modo
Chiamata
Ritorno
Tipo
Booleano
Analogico
Reale
Timer
Messaggio
Inserisci
Elimina
Disponi

10.1.13 Application example: WORKBENCH

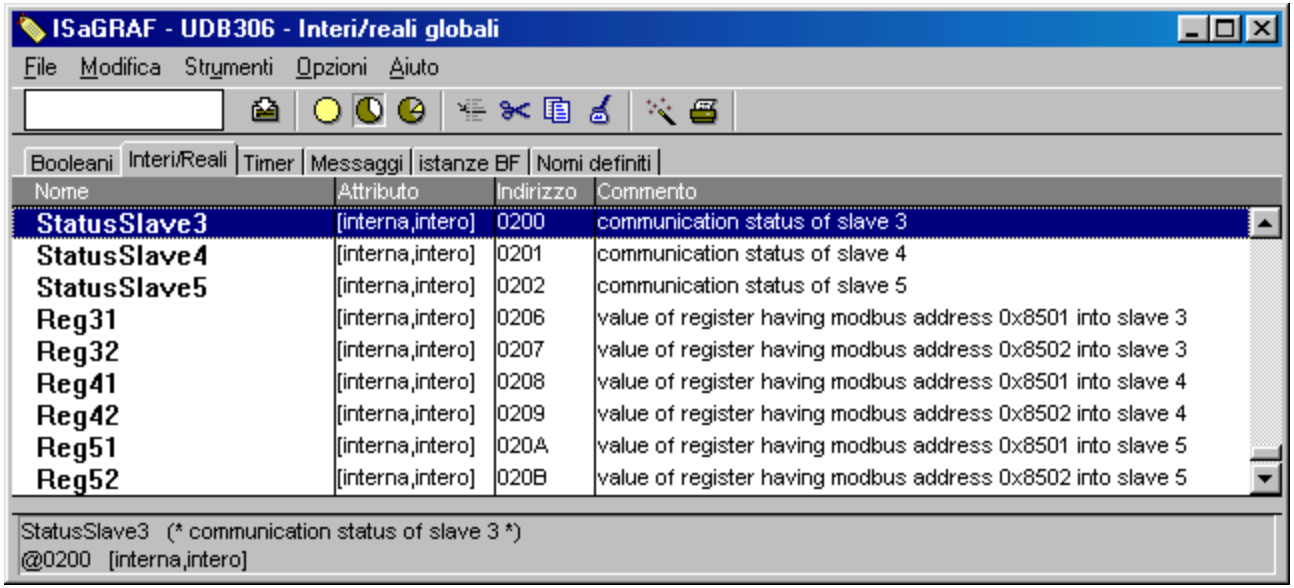
The example below uses the C-functions described above.

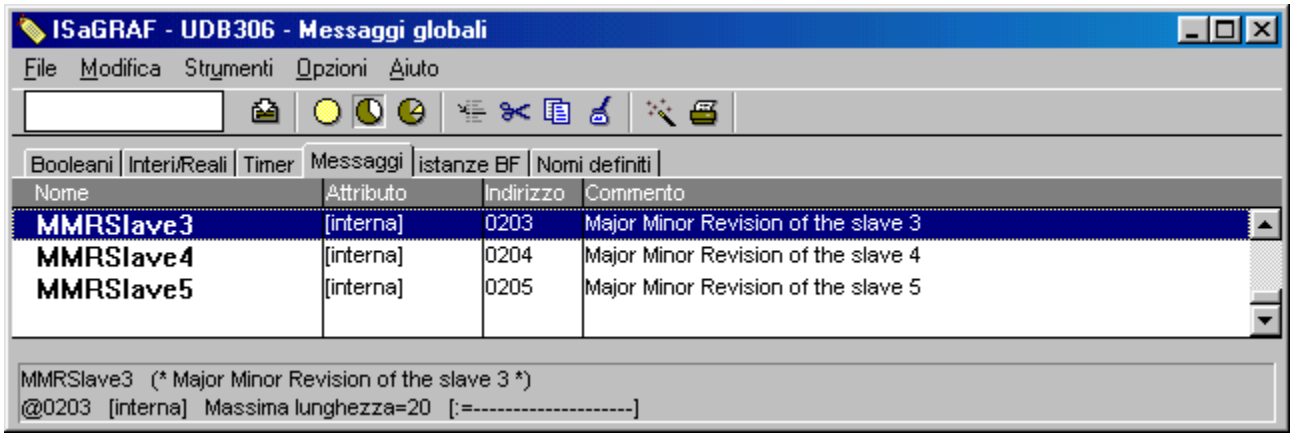
The following example refers to project:

Reference : Udb306
Author : Eliwell
Date of Creation : 31/01/2005
Version number : 3.06
Descrizione : Energy XTPRO USER sample. It arises from Udb300 with *Modbus Master* management

Let's assume that Energy XTPRO is the Master of a network with two equivalent slave devices with addresses 3, 4 and 5, connected to the *COM3* serial port. Energy XTPRO is expected to *use* these two slave devices for identification purposes, and to monitor all the salves of the two registers with *Modbus* address 0x8501 and 0x8502, and change, if required, the value of the two registers with *Modbus* address 0x18C and 0x18D for the device with serial address 4 only.

To allow the application to correctly manage the required communication, it is necessary to define the necessary variables in the library, as explained in the chapters above.





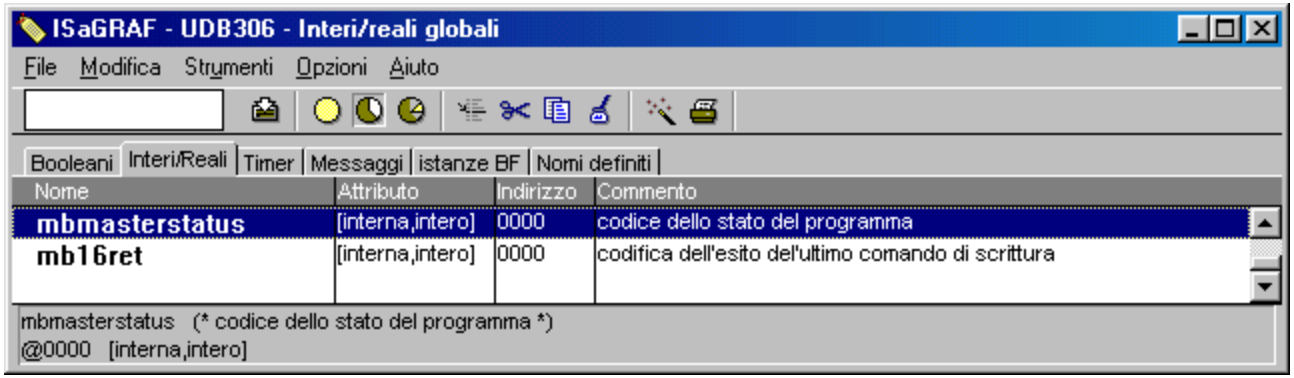
```
CASE mbmasterstatus OF

      0:
            (* this line of ST code means: on COM3 there are 3 slaves to monitor   *)
            (* with ID=3, ID=4, ID=5. Their communication status will be copied    *)
            (* in WORKBENCH's integer dictionary starting from modbus address 16#200 *)
            mbret := mbnetdef(3,16#200,3,3);

            (* go in "wait for command" status *)
            mbmasterstatus := 7;

      1: (* Command 43 request *)

            (* check line status *)
     IF not(mblnbusy(3)) THEN

                  (* this line of ST code means: on COM3 request to read one shot on slave      *)
                  (* with ID=3 , ID=4 and ID=5 the identification object (0x00)                   *)
                  (* and copy their values in WORKBENCH's message dictionary starting from modbus *)
                  (*     address 16#203. Num retry 3 and response timeout 200ms                    *)
                  mbret := mb43req(3,3,200,16#203,3,3,0);

                  IF mbret THEN

                        (* go in "wait read session complete" status *)
                        mbmasterstatus := 3;

                  ELSE

                        (* go in "wait for command" status *)
                        mbmasterstatus := 8;

                  END_IF;

            ELSE

                  (* go in "wait for command" status *)
                  mbmasterstatus := 8;

      END_IF;

      2: (* Command 03 request NOT in continuous way *)

            (* check line status *)
     IF not(mblnbusy(3)) THEN
```

```
              (* this line of ST code means: on COM3 request to read NOT in continuous way on              *)
              (* slave with ID=3, ID=4 and ID=5 two registers from 16#8501 to 16#8502 and copy their values *)
              (* in WORKBENCH's integer dictionary starting from modbus address 16#206. Num retry 3          *)
              (* and response timeout 200ms *)
              mbret := mb03req(3,3,200,FALSE,16#206,3,3,16#8501,2);

              IF mbret THEN

                    (* go in "wait read session complete" status *)
                    mbmasterstatus := 3;

              ELSE

                    (* go in "wait for command" status *)
                    mbmasterstatus := 8;

              END_IF;

       ELSE

              (* go in "wait for command" status *)
              mbmasterstatus := 8;

END_IF;

3: (* "wait read session complete" *)

       (* check line status *)
IF not(mblnbusy(3)) THEN

              (* communication session ended: now slave status can be checked *)

              (* go in "wait for command" status *)
              mbmasterstatus := 7;

       END_IF;

4: (* Command 16 request *)

       (* check previous write session progress *)
       mb16ret := mbwrres(3);

       (* previous write session ended? *)
IF mb16ret <> 4  THEN


              (* this line of ST code means: on COM3 request to write one shot on slave    *)
```

```
              (* with ID=4 two registers from 16#18C to 16#18D with these values: 0 and 30 *)
              (* Num retry 3 and response timeout 200ms *)

              mbret := mb16req(3,3,200,4,16#18C,2,0,30,0,0,0,0,0,0,0,0);

              (* go in "wait write session complete" status *)
              mbmasterstatus := 5;

       ELSE

              (* go in "wait for command" status *)
              mbmasterstatus := 8;

       END_IF;

 5: (* "wait write session complete" *)

       (* check write session progress *)
       mb16ret := mbwrres(3);

       (* write session ended? *)
IF mb16ret <> 4  THEN

              (* communication session ended: response can be checked *)

              (* go in "wait for command" status *)
              mbmasterstatus := 7;

       END_IF;

 6: (* Command 03 request in continuous way *)

       (* this line of ST code means: on COM3 request to read in continuous way on slave       *)
       (* with ID=3, ID=4 and ID=5 two registers from 16#8501 to 16#8502 and copy their values *)
       (* in WORKBENCH's integer dictionary starting from modbus address 16#206. Num retry 3          *)
       (* and response timeout 150ms *)
       mbret := mb03req(3,3,200,TRUE,16#206,3,3,16#8501,2);


       IF mbret THEN

              (* go in "wait for command" status *)
              mbmasterstatus := 7;

       ELSE

              (* go in "wait for command" status *)
```

```
                  mbmasterstatus := 8;

            END_IF;


      7: (* "wait for command" : send a new one *)
      8: (* "command aborted"  : line busy      *)

      9: (* exit form modbus master mode *)
            mbret := mbexit(3);

      (* go in "wait session complete" status *)
            mbmasterstatus := 3;

 10:  (* slave in a modbus master net *)

     IF mbslave(3,5) THEN

                  (* go in "wait for communication timeout" status *)
                  mbmasterstatus := 3;

            ELSE

                  (* go in "wait for command" status *)
                  mbmasterstatus := 8;

            END_IF;


END_CASE;
```

The mbmasterstatus variable can be used to enable the functions (read, write and mode) and monitor the status of the line. The mb16ret variable can be used to read the code of the result of the last write command.



When XTPRO is started, the Master is set to serial port *COM3*. This enables to send read and write commands, and to read the status of the communications established with slaves.
It is possible to disable the *Modbus* mode and return to it at a later stage. In addition to the Master *Modbus* mode, it is possible to enable the mbslave function that can be used to add functions to the slave mode.

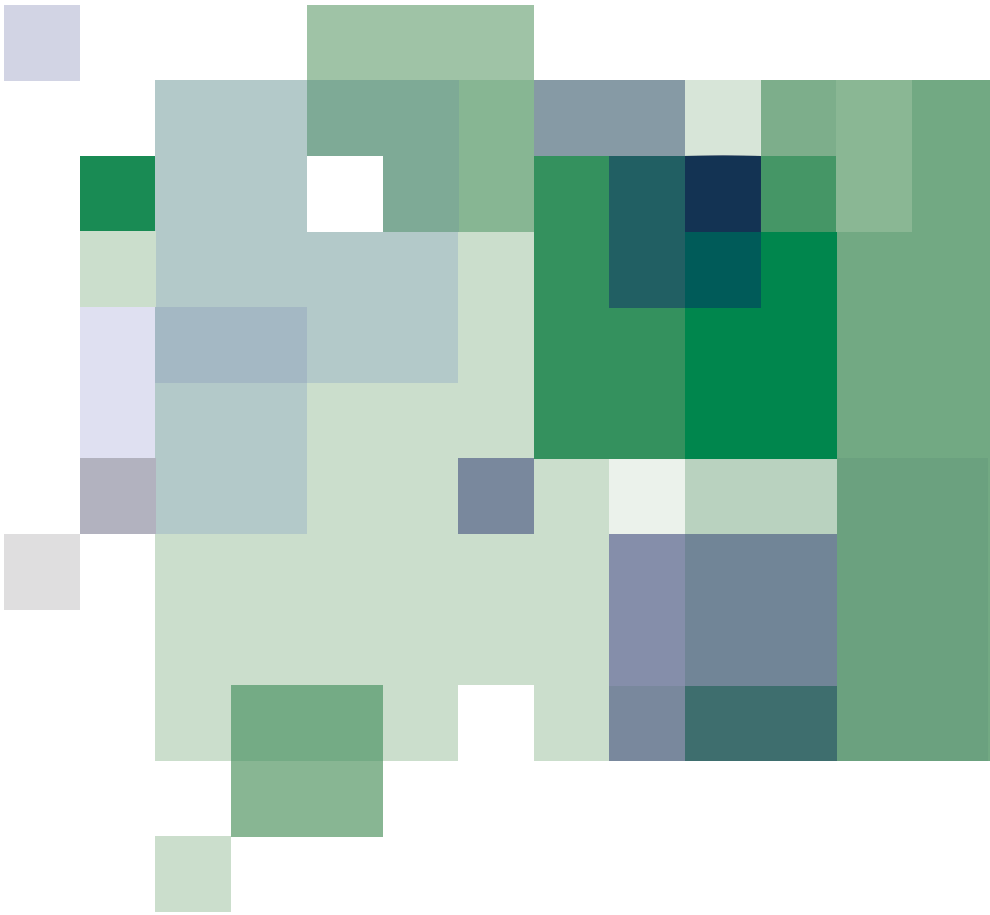
11 ANALTIC INDEX

Eliwell Controls S.r.l.
Via dell' Industria, 15 Zona Industriale Paludi
32010 Pieve d' Alpago (BL) Italy
Telephone +39 0437 986 111
Facsimile +39 0437 989 066

Sales:
+39 0437 986 100 (Italy)
+39 0437 986 200 (other countries)
saleseliwell@invensyscontrols.com

Technical helpline:
+39 0437 986 300
E-mail techsuppeliwell@invensyscontrols.com

www.eliwell.it



EXT PRO Communication Protocols
2009/02/
Cod: 8MA00055