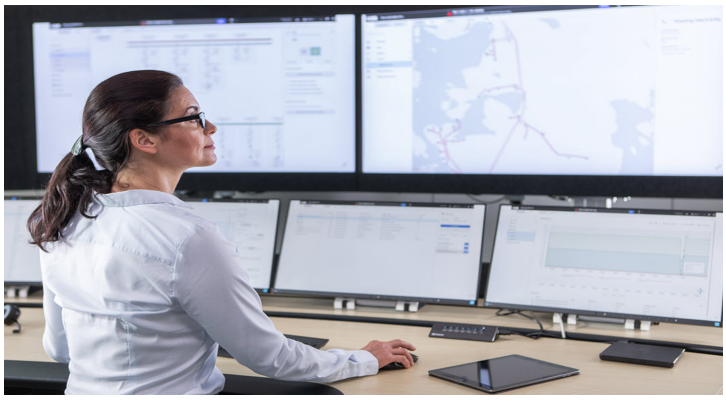
GRID AUTOMATION PRODUCTS

MicroSCADA X SYS600 10.2

Modbus Master Protocol



Document ID: 1MRK 511 497-UEN
Issued: March 2021
Revision: A
Product version: 10.2

Table of contents

Section 1 Copyrights

The information in this document is subject to change without notice and should not be construed as a commitment by Hitachi Power Grids. Hitachi Power Grids assumes no responsibility for any errors that may appear in this document.

In no event shall Hitachi Power Grids be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall Hitachi Power Grids be liable for incidental or consequential damages arising from the use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without written permission from Hitachi Power Grids, and the contents thereof must not be imparted to a third party nor used for any unauthorized purpose.

The software or hardware described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

Trademarks

ABB is a registered trademark of ABB Asea Brown Boveri Ltd. Manufactured by/for a Hitachi Power Grids company. All other brand or product names mentioned in this document may be trademarks or registered trademarks of their respective holders.

Guarantee

Please inquire about the terms of guarantee from your nearest Hitachi Power Grids representative.

Third Party Copyright Notices

List of Third Party Copyright notices are documented in "3rd party licenses.txt" and other locations mentioned in the file in SYS600 and DMS600 installation packages.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (https://www.openssl.org/). This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Section 2 Introduction

2.1 This manual

This manual is a programming manual. It describes the principles of interfacing SYS600 to an external device using the Modbus protocol.

In order to fully understand the concepts outlined here, the reader should be familiar with the SCIL programming language and general SYS600 application techniques. General knowledge about the Modbus protocol and PLC programming is also needed.

2.1.1 Modbus master protocol

The Modbus master protocol is mainly used for a master-slave connection between intelligent devices. This means, in our case, connecting Programmable Logic Controllers (PLCs) to SYS600.

2.1.2 General principles of the Modbus protocol

The Modbus Communications Protocol is an asynchronous, byte packaged protocol used for communications between the master stations and Intelligent Electronic Devices (IEDs) or Remote Terminal Units (RTUs). It provides a transport mechanism for the master's requests and RTU response messages. It supports one single master station and up to 247 RTUs on a multi-drop line. The Modbus protocol functions as a true serial line or Transmission Control Protocol (TCP) network.

The Modbus protocol has two distinct modes: ASCII Modbus, which uses ASCII-encoded hexadecimal messages and binary Modbus, which uses raw binary messages, see SYS600 Modbus Slave Protocol for more information.

All transactions are initiated by transmission of a request from the master station, an RTU may not transmit unsolicited information. Every master station request must be addressed to a specific RTU and some implementations of Modbus do not support the broadcast message request type. A transaction consists of a single master station request, followed by an RTU response or exception frame or a master station timeout if no RTU response is generated.

There are many different kinds of devices, which use the Modbus protocol for communication. Therefore, the SYS600 Modbus protocol emulation is somewhat restricted. All the features that are available in external devices are not necessarily available in the NET implementation.

The general strategy that is used in this implementation can be summarized as follows:

- PC-NET contains necessary protocol conversion features, which enable it to send and receive Modbus telegrams.
- The SCIL application program in the base system sees the Modbus implementation in the PC_NET as a PLC type device.
- The PLC type device uses the SYS600 process database like a RTU device.
- Cross-reference information of correlation between the RTU addresses and Modbus addresses is stored in the topic configuration data of the PLC device
- A SYS600 application program can refer to an external device data through the SYS600 process database, or it can use direct communication attributes to read from or write data to an external device.

- PC-NET is the protocol master. The communication with external devices is done by using the Modbus RTU protocol mode.
- The current version of the Modbus implementation in the PC_NET supports functions 1, 2, 3, 4, 5, 6, 15, 16.
- There is also a limited support for functions 20 and 21. These functions are used for event table transmission and time-synchronization for the REF 542plus protection relay.

2.1.3 Modbus TCP master protocol

The Modbus TCP master protocol is used in LAN and WAN networks to connect central stations and outstations to each other. Since the stations use an open TCP/IP interface as a connection to the network, the structure and the characteristics of the network will be invisible to the SYS600 communication software. The overall performance of Modbus/TCP is higher compared to serial line Modbus due to the faster transmission speed.

The SYS600 implementation of Modbus TCP operates as a master. The protocol numbers that are used are equal to the current ones. The SD attribute of the line defines if the line uses serial port or TCP. The SD attribute must be set before the line is taken into use for the first time.

Examples

```
#SET NET'NET':SSD'LINE' = "COM5 ";line uses serial port 5
#SET NET'NET':SSD'LINE' = "TCP ";line uses TCP connection
```

SYS600 is able to keep several connections open to the controlled stations at the same time. Multiple Modbus/TCP lines may be created in the same computer.

The transferred data messages are close to the messages used in serial line based Modbus protocol. Compared with serial line implementation, a seven byte MBAP header is added to the beginning of each message. The Modbus master operates always as a TCP client.

For more information on Modbus TCP, see the following web site: www.modbus.org: Modbus messaging on TCP/IP.

Using line attribute OM bit 1, it is possible to configure the Modbus TCP line to operate in 'Serial over IP' mode where the MBAP header is not used and messaging is similar to serial mode. See description of line attribute OM for more information.

2.2 Use of symbols

This publication includes warning, caution and information symbols where appropriate to point out safety-related or other important information. It also includes tips to point out useful hints to the reader. The corresponding symbols should be interpreted as follows:

Warning icon indicates the presence of a hazard which could result in personal injury.

Caution icon indicates important information or a warning related to the concept discussed in the text. It might indicate the presence of a hazard, which could result in corruption of software or damage to equipment/property.

Information icon alerts the reader to relevant factors and conditions.

Tip icon indicates advice on, for example, how to design a project or how to use a certain function.

Although warning hazards are related to personal injury, and caution hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all warnings and caution notices.

2.3 Related documents

The following SYS600 manuals should be available for reference during the use of this manual:

Name of the manual	Document ID
SYS600 10.2 System Configuration	1MRK 511 481-UEN
SYS600 10.2 System Objects	1MRK 511 482-UEN
SYS600 10.2 Application Objects	1MRK 511 467-UEN
SYS600 10.2 Status Codes	1MRK 511 480-UEN

Other related documents:

- Product documentation of the used modem

2.4 Document revisions

Revision	Version number	Date	History
A	10.2	31.3.2021	New document for SYS600 10.2

Section 3 Safety information

This section has information on the prevention of hazards and taking backups from the system.

3.1 Backup copies

3.1.1 Taking backup copies

We recommend taking backup copies before making any changes, especially ones that might have side effects. Software and data need to be copied to another place.

Backup copying makes it easier to restore the application software in case of disk crash or other severe failure where stored data is lost. It is therefore recommended that backup copies are taken regularly.

There should be at least two system backup copies and two application copies. A new backup is copied over the oldest backup. This way the latest version is always available, even if the backup procedure fails.

Detailed information on how to take backup copies should be delivered to the customer with the application.

3.1.2 System backup

Usually a system back up is taken after the application is made. It should be taken again when changes are made to the SYS600 system. This is required when the driver configuration or the network setup is changed.

3.1.3 Application backup

An application backup is also taken at the same time with the system backup, after the application is made. It should be taken again when changes are made to the application, for example, if pictures or databases are edited or new pictures are added.

3.2 Fatal errors

A fatal error is an error that causes a breakdown or a locked situation in the SYS600 program execution.

3.2.1 Handling

In case of a fatal error:

1. Write down the possible SYS600 error messages.
2. Shut down the SYS600 main program. If this cannot be done in the SYS600 Control Panel, try to end the task in Windows Task Manager.

Files may be damaged if the base system computers are shut down by switching the power off.

3. The data kept in the main memory at the moment of a fatal error is placed in the drwtsn32.log file with Windows 2003 Server, Windows XP and earlier. By default it is placed under %SYSTEMDRIVE%\Documents And Settings\All Users\Application Data \Microsoft\Dr Watson. Log and dump file paths can be checked with the drwtsn32 application. (Start -> run -> drwtsn32.exe). Analyze and copy the data in these files. Starting with Windows Server 2008 and Windows 7 the crash handling has changed. The location of the dump files can be read from the registry under the key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting \LocalDumps. The DumpFolder value tells the location of the dump files. Collect the data from this location.
4. Restart the system.

Report the program break-down together with the possible SYS600 error messages and the information from the drwtsn32.log file to the SYS600 supplier.

3.2.2 Status codes

Error messages in SCIL are called status codes. A list of status codes and short explanations for them can be found in SYS600 Status Codes.

Section 4 Communication system configuration

Each NET unit contains a set of system objects, which specify line properties, connected devices etc. These objects can be created, modified and deleted by SCIL, and setting the attributes of the objects can change their properties.

Access to the attributes can be one of the following:

- **Read-only:** The attribute can only be read. There are still a few exceptions in which the values can be reset.
- **Write-only:** The attribute can only be written (set).
- **Read, conditional write:** The attribute can be both read and written, but the object must be set out of use (IU = 0) before writing.
- **No limitations:** The attribute can be both read and written without limitations.

The implementation of the Modbus master protocol in SYS600 can be divided into two layers: link layer and application layer. Both of these layers have a specific functionality and a set of attributes of their own. The link layer corresponds to a line of a NET unit and the application layer corresponds to a station configured to the line.

The purpose of the communication system configuration is to:

- Create all the system objects needed to establish communication between the master and the slave.
- Adjust the values of the system object attributes to match the physical communication channel and the properties of the slave station.

4.1 Setting the attribute values

All line and station attributes have sensible default values but the value of each attribute must be checked against the requirements of the real communication system. The attribute values depend on:

- The physical communication media (for example leased telephone line, radio link, and power line carrier). This particularly affects the attributes of the line, such as baud rate and parity.
- The network topology used (point-to-point, multidrop), which affects the link type.
- The size (number of stations) of the system, which especially affects the timeout parameters: the slower the media and bigger the system, the longer the timeouts that are needed.

4.2 Network topologies

The implementation of the Modbus master protocol in SYS600 supports direct and serial bus topologies. The direct topology (point-to-point) can be a direct physical cable from point-to-point or a two-node radio, or modem network. The serial bus topology (multi-drop) is commonly made up of many modems with their outputs and inputs either tied together or linked using a star-coupler.

The protocol can be used with virtual serial ports with a special setting of the line attributes. See NET line attribute CM for more information.

The Modbus protocol supports one master on a line. The following figure illustrates the network topologies:

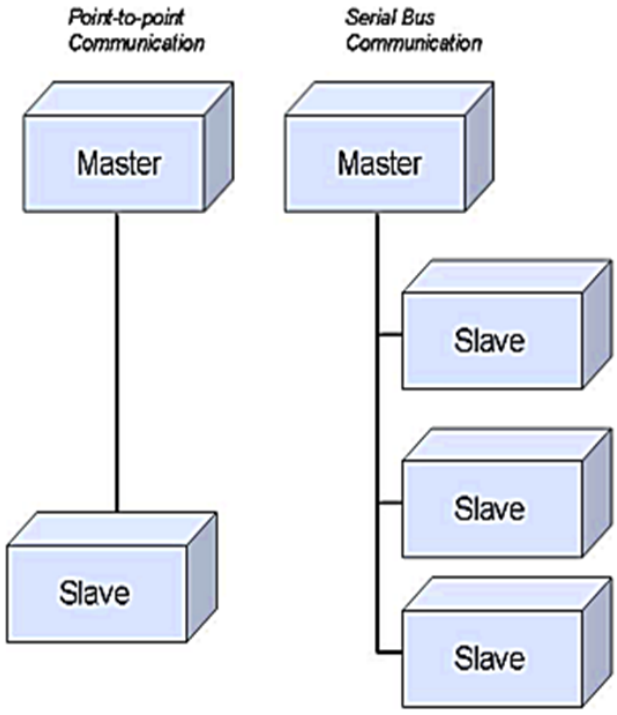


Figure 1: Network topologies

4.3 Modbus line configuration

The line process of a NET unit performs the functions of the link layer. The purpose of the link layer is to send messages to and receive them from external devices using the Modbus protocol. The link layer also provides frame synchronization and link control.

4.3.1 Line attributes

The following attributes can be used to configure the Modbus master lines in SYS600.

IU	In Use
Indicates whether the line is in use (value 1) or not in use (value 0).	
Data type:	Integer
Value:	0 or 1
Index range:	1...12 (NET line numbering)
Default value:	0
Access:	No limitations

PO	Protocol

The data transfer protocol used on the line. The line is defined to the NET by setting this attribute. By setting the attribute to 0 the line definition including all the line attributes is deleted.

Data type:	Integer
Value:	0... 35 Value with Modbus Master protocol: 25
Index range:	1...12 (NET line numbering)
Access:	Read, conditional write

SD	System Device Name

Associates the NET line numbers of PC-NET with the device names of the physical channels of the serial ports. By default, line number 1 is connected to COM1, line 2 to COM2 and so on. It is possible to override these default values by using the SD attribute . This may be necessary if COM ports are used as NET lines, or if a RocketPort card or similar is used. If the line should use Modbus TCP/IP, the string "TCP" must be given. See also Section Modbus TCP/IP master attribute under Section 4.4

Data type:	Text
Value:	See above
Index range:	1...12 (NET line numbering)
Access:	Read, conditional write

Example:

```
#SET NET'NET':SSD'LINE' = "COM5";line uses serial port 5
#SET NET'NET':SSD'LINE' = "TCP";line uses TCP connection
```

PS	Buffer Pool Size

Specifies the number of message buffers reserved for the line. Fixed buffer poll sizes are used in versions 9.3 FP1 and newer and this attribute is retained because of the backward compatibility. Setting the value for PS is not possible anymore. See the attributes PS, NB and PB from the System Objects manual for more information.

Data type:	Integer
Value:	1... 250
Index range:	1...12 (NET line numbering)
Access:	Read (conditional write accepted but has no effect)

BR	Baud Rate

Transmission rate used on the line. This attribute is meaningless if the line operates in the TCP/IP mode.

Data type:	Integer
Value:	1...19200 (384 = 38400 bauds, 576 = 57600 bauds)
Unit:	Bits/s
Index range:	1...12 (NET line numbering)
Default value:	9600
Access:	Read, conditional write

PY	Parity

Specifies the parity check (if any) used for the characters transferred on the line. This attribute is meaningless if the line operates in the TCP/IP mode.

Data type:	Integer
Value:	0 = no parity check 1 = odd parity 2 = even parity

Table continues on next page

PY	Parity
Index range:	1...12 (NET line numbering)
Default value:	0
Access:	Read, conditional write

RD	Receiver Data Bit Count

Specifies the number of data bits in each received character. This attribute is meaningless if the line operates in the TCP/IP mode.

Data type:	Integer
Value:	5, 6, 7 or 8
Unit:	Data bits
Index range:	1...12 (NET line numbering)
Default value:	8
Access:	Read, conditional write

SB	Stop Bits

Specifies the number of stop bits attached to each transmitted character. This attribute is meaningless if the line operates in the TCP/IP mode.

Data type:	Integer
Value:	1 or 2
Unit:	Stop bits
Index range:	1...12 (NET line numbering)
Default value:	1
Access:	Read, conditional write

TD	Transmitter Data Bit Count

Specifies the number of data bits in each transmitted character. This attribute is meaningless if the line operates in the TCP/IP mode.

Data type:	Integer
Value:	5, 6, 7 or 8
Unit:	Data bits
Index range:	1...12 (NET line numbering)
Default value:	8
Access:	Read, conditional write

CM	COM Port Mode

This attribute consists of a set of flags which control the behavior and functionality of the serial port of the line. Each flag is one bit of this attribute. This attribute is meaningless if the line operates in the TCP/IP mode.

Data type:	Integer
Value:	0...15 (see below)
Index range:	1...12 (NET line numbering)
Default:	0
Access:	Read, conditional write

Table continues on next page

CM	COM Port Mode
Bit 0:	UART error handling When this bit is 0, the UART errors are read before the bytes are read from the serial port. This is the default mode. When the bit is 1, the UART errors are read as a separate operation after the bytes are read from the serial port. This mode is similar to PC_NETs older than 9.2SP2 and it does not detect all errors detected by the serial port hardware. If the line has a lot of disturbances, this mode may result in better performance than the default mode.
Bit 1:	Simulated 'high' of the CTS signal When this bit is 0, the actual state of the CTS signal is used in the protocol. This is the default mode. When this bit is 1, the CTS signal is simulated to be in 'high' state all the time and the line behaves accordingly. This setting may be necessary with a virtual serial port or for easier cabling. See line attribute DE how transmission starts when CTS is constantly 'high'.
Bit 2:	Simulated 'high' of the DCD signal When this bit is 0, the actual state of the DCD signal is used in the protocol. This is the default mode. When this bit is 1, the DCD signal is simulated to be in 'high' state all the time and the line behaves accordingly. This setting may be necessary with a virtual serial port or for easier cabling.
Bit 3:	Calculated RTS keep up time When this bit is 0, the keep up time of the RTS signal is not calculated using the length of the message but it is assumed that the driver of the serial port blocks the execution of the sending process until the message is actually sent. This setting should be used if the setting 'Wait on physical transmission before completing write' or similar is selected in the driver for the port in question. The tuning of the RTS keep up time should be done with line attribute RY. This is the default setting When this bit is 1, the keep up time of the RTS signal is calculated using the length of the sent message and the baudrate of the port. The RTS keep up time defined with the line attribute RY is added to the calculated time. This setting is not needed if the setting 'Wait on physical transmission before completing write' or similar is selected in the driver for the port in question. This setting is the most common with the virtual serial ports, too. If the serial driver does not provide setting 'Wait on physical transmission before completing write' or similar and RTS signal is actively used by the modem hardware, it is worth to test both alternatives. For accurate analysis using protocol analyzer function, see also the description of the bit 2 of the line attribute AU Analyzer Usage.

Having a simulated value in CTS or DCD may have an effect on how an RS-232 line disconnection is detected and reported to the MicroSCADA application.

PD	Polling Delay
Delay between polling messages. The master sends the request with an interval defined by this attribute.	
Data type:	Integer
Value:	0...65535
Unit:	Milliseconds
Index range:	1...12 (NET line numbering)
Default value:	40
Access:	Read, conditional write

DE	CTS Delay

Time delay (in milliseconds) between the activation of the RTS signal (Request to Send) and the start of a new transmission. Transmission starts if CTS is high after the delay or line attribute CM, bit 1, is set (serial mode, CTS simulated to be constantly high). If CTS stays low and CM bit 1 is not set, CTS error is reported. In practise, this occurs when line is disconnected. With DE value 0, maximum waiting time is constant 500 msecs but the transmission starts immediately when the CTS signal is detected to be signaled (rising edge). If CTS is constantly signaled or CM bit 1 is set and DE=0, each transmission is delayed with 500 ms. With Modbus TCP/IP, each message transmission is delayed with DE milliseconds.

Data type:	Integer
Value:	0...65535
Unit:	Milliseconds
Index range:	1...12 (NET line numbering)
Default value	0
Access:	Read, conditional write

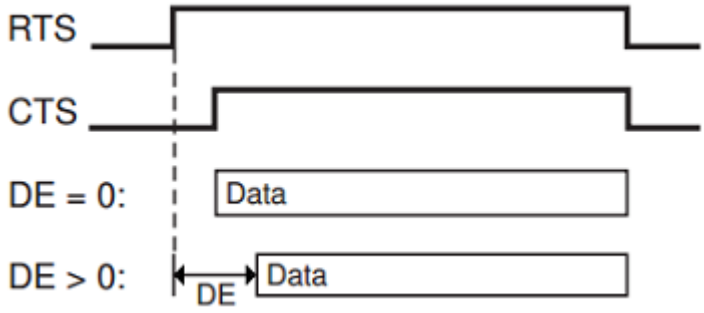


Figure 2: DE attribute

HT	Header Timeout

Specifies the maximum waiting time in milliseconds within which the first byte of a link layer response should have been received after the transmission of a message. If no response has been received within this time, new attempts are performed the number of times specified by the Enquiry Limit. If no response is still obtained, the station is be suspended.

Data type:	Integer
Value:	0...65535
Unit:	Milliseconds
Index range:	1...12 (NET line numbering)
Default value:	700
Access:	Read, conditional write

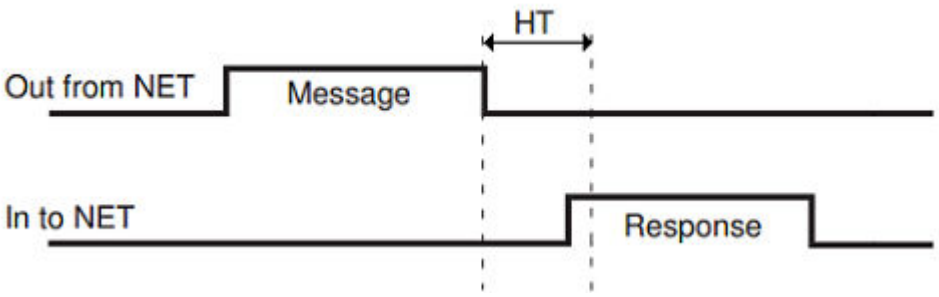


Figure 3: HT attribute

TI	Response Timeout

The time in seconds that the Modbus link layer waits for the end of the response message. This timeout is internally limited to 32 seconds.

Data type:	Integer
Value:	0...255
Unit:	Seconds
Index range:	1...12 (NET line numbering)
Default value:	2
Access:	No limitations

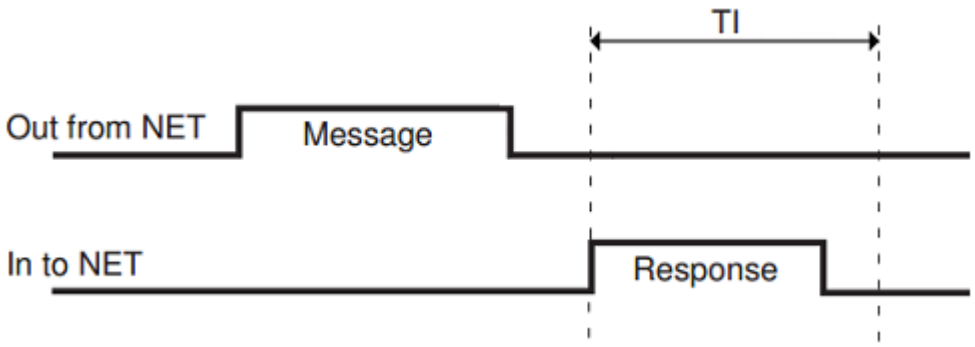


Figure 4: TI attribute

RI	Receive Interrupt Enable Delay

Defines the delay in milliseconds after which the receiver of a NET line is enabled when a message has been issued.

Data type:	Integer
Value:	0...255 0 = receiver enabled all the time 1…9 = receiver enabled right after transmission 10… = receiver enabled as stated by the value
Unit:	Milliseconds
Index range:	1...12 (NET line numbering)
Default value:	5 (Serial), 0 (TCP)
Access:	Read, conditional write

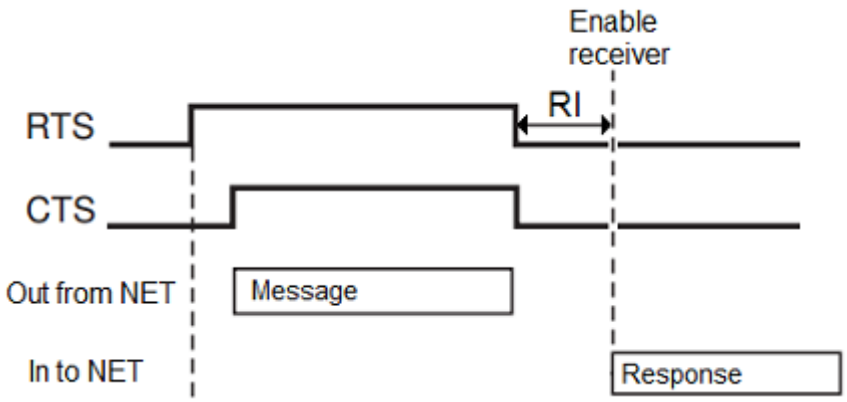


Figure 5: RI attribute

RY	RTS Keepup Delay

This attribute defines how long time the RTS-pin of the RS232-port is kept in the signal state after the serial driver completes the write operation. The write operation here means a transmission of any message. See also line attribute CM (Com Port Mode).

Data type:	Integer
Value:	0...20
Unit:	Bytes (absolute time depends on baudrate)
Index range:	1...12 (NET line numbering)
Default value:	1
Access:	Read, conditional write

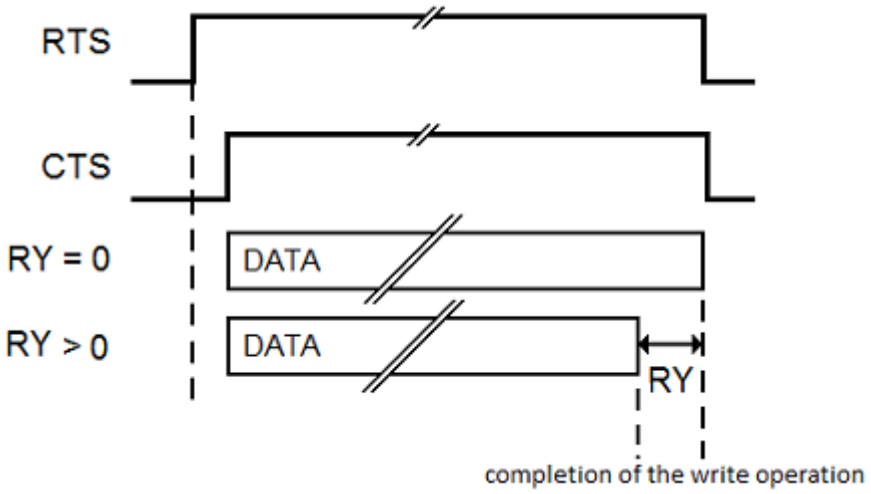


Figure 6: RY attribute

EN	Enquiry Limit

Specifies the maximum number of times that a message is retransmitted after a timeout.

Data type:	Integer
Value:	1...255
Index range:	1...12 (NET line numbering)
Default value:	6
Access:	Read, conditional write

SG	Modem Signal

An attribute for direct supervision and control of the state of the modem signal. This attribute applies to all protocols. It is used for diagnostics and testing.
If the incoming signal DCD or CTS is wanted to have a simulated 'high' value all the time, value = 1 can be written to these signals. This feature may be necessary for easier cabling or with virtual serial ports. If value = 0 is written to these signals, the actual state of signal will be used. The default mode of operation is the actual state. See also the attribute CM Com Port Mode.

Data type:	Integer
Value:	0 ,1 0 = Passive signal 1 = Active signal
Incoming:	DCD and CTS signals
Outgoing:	DTR signal

Table continues on next page

SG	Modem Signal
Index range:	100 * line nr + signal no. Signal no. 5 = CTS, 8 = DCD, 20 = DTR
Access:	Read-only, write possible to signals 5 = CTS and 8 = DCD

Examples:

```
#SET NET1:SSG208 = 1 ; line 2 of NET1 should behave as DCD is 'high'
all the time
#SET NET1:SSG205 = 1 ; line 2 of NET1 should behave as CTS is 'high'
all the time
#SET NET1:SSG208 = 0 ; line 2 of NET1 should use the actual state of
the DCD
#SET NET1:SSG205 = 0 ; line 2 of NET1 should use the actual state of
the CTS
```

MI	Message Identification
Object address of system messages.	
Data type:	Integer
Value:	1... 32760
Index range:	1...12 (NET line numbering)
Default value:	6000 + (100 * NET number) + line number
Access:	Read, conditional write

MS	Message Application
The number of the application that is the receiver of the system messages generated by the line.	
Data type:	Integer
Value:	1... 250
Default value:	1
Index range:	1...12 (NET line numbering)
Access:	Read, conditional write

PM	Protocol Mode
The type of the Modbus protocol in use.	
Data type:	Integer
Value:	0, 1 0: RTU 1: ASCII
Index range:	1...12 (NET line numbering)
Default value:	0
Access:	Read, conditional write

OM	Operating mode
A bit pattern which defines the operating mode of the line.	
Data type:	Integer
Value:	0..65535 (see below)
Index range:	1...12 (NET line numbering)
Default:	0
Access:	Read, conditional write

Table continues on next page

OM	Operating mode
Bit 0:	Transaction ID Checking. When this bit is set to 1, checking for the transaction identification number of the responses is disabled. The checking has to be disabled if some of the Modbus TCP devices connected to the line do not support transaction identification copying from the request to the response. In this configuration, there is a risk that a delayed response is interpreted incorrectly. Therefore, the usage of the link layer retries defined with line EN is not recommended. When this bit is not set, the checking of the transaction identification number of the responses is enabled. Remote devices must copy the transaction identification from the request to the corresponding response. This is the default mode of operation. See also OM bit 3, One byte Transaction ID checking.
Bit 1:	Serial over IP mode. If this bit is set, all messaging of the line is similar to serial Modbus i.e. the MBAP message header used in Modbus TCP is not included to messages. This setting has an effect on all STA objects connected to the line. Otherwise, the configuration is similar to normal Modbus TCP. Serial over IP mode can be used if the remote device does not support standardized Modbus TCP or the serial Modbus is tunneled to TCP/IP as such. Mode is available both in RTU and ASCII modes. If this bit is not set, MBAP header is included to the messages and functionality is according to Modbus TCP Standard (RTU mode). This is the default mode of operation.
Bit 2:	CRC checking in serial over IP mode. If this bit is set, CRC (Cyclical Redundancy Check) value of the incoming message need not to be correct or it need not to present at all. Setting of this bit may useful if the gateway device is doing the CRC checking. If this bit is not set, CRC of the incoming message must be correct similarly to serial mode of Modbus. This is the default mode of operation.
Bit 3:	One byte Transaction ID checking. When this bit is set to 1, checking for the transaction identification field is made only for the lowest byte of the two-byte transaction ID field. This checking has to be enabled if the Modbus TCP device has an upper limit of 255 in its transaction identification values in responses to SYS600. A modbus response which has incorrect transaction ID is not processed. If transaction ID is totally disabled using OM bit 0, the value of this bit has no effect. If this bit is not set and OM bit 0 is not set either, the checking of the transaction identification is made using both bytes as defined in standard. This is the default mode of operation.

LK	Link Type

The type of the data link connection on the line.
The states of the CTS and DCD signals of the serial may have simulated values. The usage of this feature may be necessary if the line uses a virtual serial port, or if the hardware connected to the serial port requires a special cable. See the description of the line attribute CM for more information.

Data type:	Integer
Value:	1 = Modem line. A normal RTS-CTS modem signal handshaking is applied 16 = No RTS-CTS handshaking Other values are accepted as well but the behavior is similar to value 1
Default value:	1
Index range:	1...12 (NET line numbering)
Access:	Read, conditional write

CB	Carrier Blocking

This attribute determines whether the Carrier Detect (DCD) signal of the serial port must be set in order for the Modbus line to receive messages. The DCD pin of the serial port is used for this attribute.

Data type:	Integer
Value:	0 = Carrier blocking not used, Carrier Detect ignored 1 = Carrier blocking not used, Carrier Detect must be set
Default value:	1
Index range:	1...12 (NET line numbering)
Access:	Read, conditional write

PP	Polling Period

The polling frequency of suspended stations. This attribute specifies how often suspended stations on the line are polled. Normally, the NET unit is continuously polling the stations according to the configuration of the topics in PLC device objects. In case some of the stations are in a suspended state, it is possible to improve the overall performance of the communication line by using the PP attribute. If PP>1, the Modbus line skips all other polls to the suspended station except for every PP'th poll. On the other hand, the communication is re-established more slowly when the PP has a big value.

Data type:	Integer
Value:	1 ... 255
Indexing:	Line number
Suggested value:	1 ... 10
Default value:	3
Access:	Read, conditional write

Example:
#SET NET3:SPP1=5 ; only every 5th poll to the suspended station is made

DC	Diagnostic Counters

The line protocols gather statistical information about events on the lines by incrementing a number of diagnostic counters. All the major events and error situations of the communication have their own counters. When accessing diagnostic counters, this attribute is indexed according to the formula:
100 * (line number) + (diagnostic counter number)
Modbus master protocol supports the following counters:
1. Transmitted telegrams
2. Failed transmissions
4. Transmitted commands
5. Transmitted replies
11. Received messages
12. Parity errors
13. Overrun errors
14. Check sum errors
15. Framing errors
16. Buffer overflow errors

Data type:	Integer
Value:	0...30000
Index range:	See above
Access:	Read-only, the values can be reset

4.3.2 Example configuration of Modbus Serial line

The following example shows the Modbus serial line configuration:

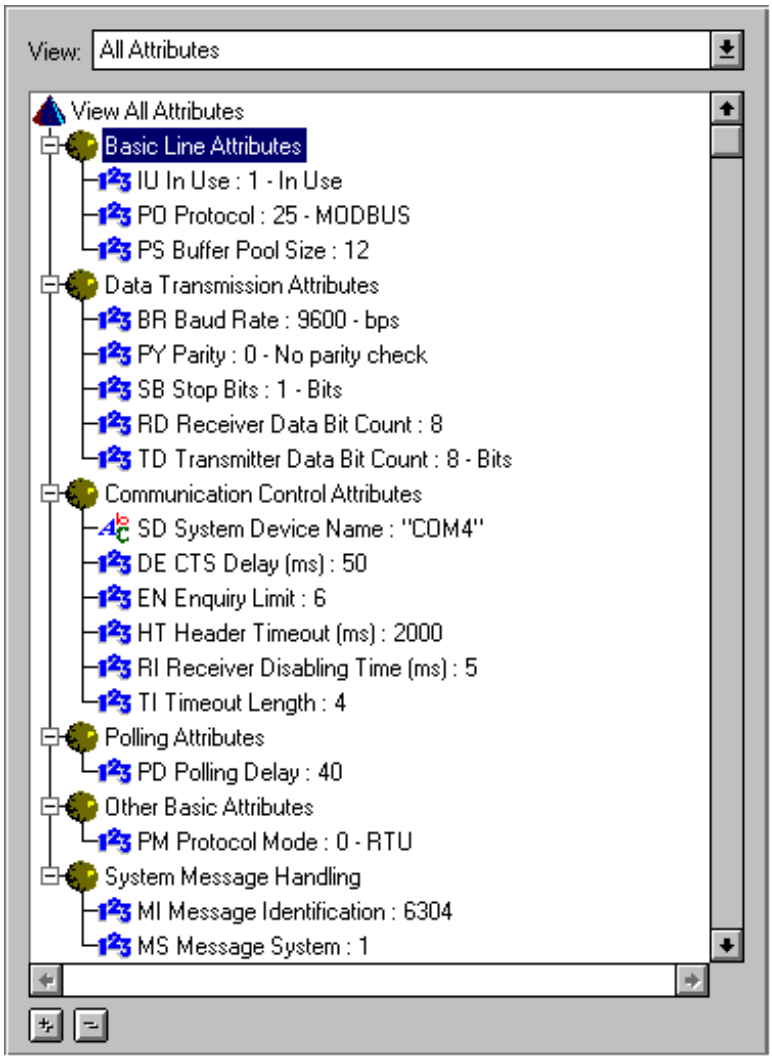


Figure 7: Example of Modbus serial line configuration

4.3.3 Modbus TCP/IP master attribute

The following attributes can be used for configuring the Modbus master/TCP lines in SYS600. The LD attribute is meaningful only if there are multiple IP addresses in the computer.

LD	Local Address

The IP address which is used locally. The setting of this attribute is necessary when the computer has multiple IP addresses and it is defined which address the created line must use. This attribute must be set before the line has been taken into use for the first time. An empty string in LD means that the default IP address of the computer is used. The value of the LD cannot be modified after the line has been taken into use for the first time.

Data type:	Text
Value:	String containing a valid IP address, max 29 characters
Default:	Empty string
Index range:	1...12 (NET line numbering)
Access:	Read/write

This attribute accepts the IP address in the following form:

Table continues on next page

LD	Local Address

```
#SET NET1:SLD1="10.0.0.1"
```

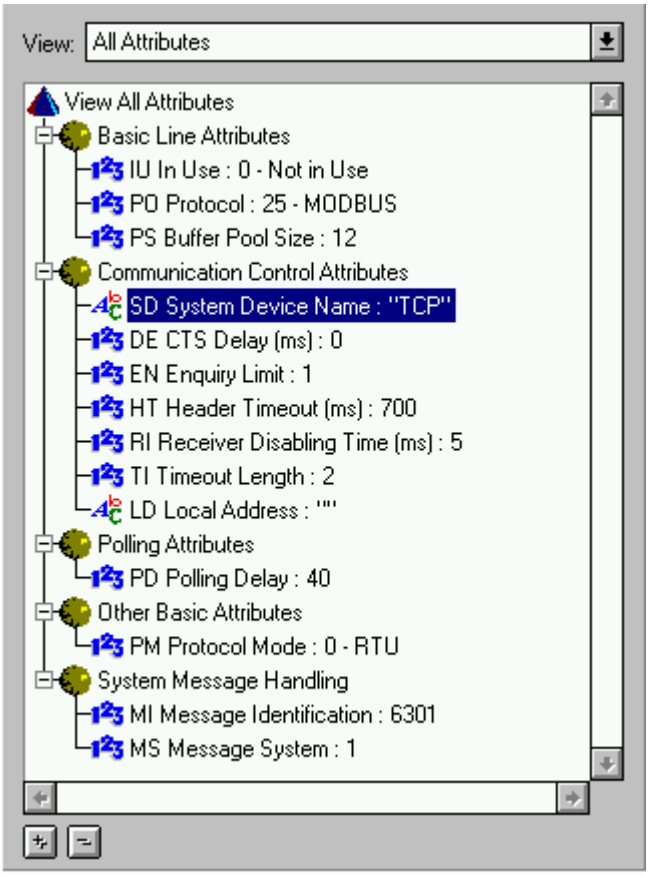


Figure 8: Example of Modbus TCP line configuration

There is an internal limitation which prohibits the use of the same local IP address and the same line number multiple times. This applies to all PC_NET protocols using LAN.
Example:
It is not possible to have
IEC104 master in Line 1 in PC_NET 1 with LD="192.168.1.1"
and
DNP3.0 master in Line 1 in PC_NET 2 with LD="192.168.1.1"
the configuration must be changed to
IEC104 master in Line 2 in PC_NET 1 with LD="192.168.1.1"
and
DNP3.0 master in Line 1 in PC_NET 2 with LD="192.168.1.1"
or to
IEC104 master in Line 1 in PC_NET 1 with LD="192.168.1.1"
and
DNP3.0 master in Line 1 in PC_NET 2 with LD="192.168.1.2"
The limitation is present only when the same local IP address is used. An easy workaround is to configure multiple IP-addresses which are using the same adapter. If this is not possible, setting a unique value for the NET node attribute LP redefines the internally used ports for the NET node so that there is no conflict. See SYS600 System Objects manual for more information about the NET Node attribute LP.

DC	Diagnostic Counters

The line protocols gather statistical information about the events on the lines by incrementing a number of diagnostic counters. All the major events and error situations of the communication have their own counters. When accessing diagnostic counters, the attribute is indexed according to the formula:
100 * (line number) + (diagnostic counter number)
Modbus master protocol supports the following counters:
1. Transmitted telegrams
2. Failed transmissions
4. Transmitted commands
5. Transmitted replies
11. Received messages
12. Parity errors
13. Overrun errors
14. Check sum errors
15. Framing errors
16. Buffer overflow errors
20. TCP connect
21. TCP accept
22. TCP close

Data type:	Integer
Value:	0...30000
Index range:	See above
Access:	Read-only, the values can be reset

4.3.4 Autodialing attributes

SYS600 supports the autocaller functionality for Modbus Master Protocol. An autocaller is a modem, which contains automatic dial-up functions. Modbus Master Protocol or Modbus Slave Protocol can call up.

The autocaller uses the AT (Hayes) command set. When using odd or even parity, the modem has to support 11-bit word length. In some cases, the autocaller is enabled with the AT commands. For more information about the modem, refer to its manual.

The following autocaller attributes are valid for the Modbus Master Protocol dial-up lines:

AC	Autocaller Enabled

States whether an autocaller is connected to the line (value 1) or not (value 0).

Data type:	Integer
Value:	0 or 1
Default value:	0
Access:	No limitations

AS	Autocaller State

Indicates the state of the autocaller.

Data type:	Integer
Value:	0...4 0 = IDLE, ready to call up 1 = CONNECTED, transmission is activated 2 = BUSY, autocaller is dialing 3 = INITIAL, autocaller is uninitialized 4 = CONFIGURE, the IU attribute of the line is set to 0
Default value:	0
Access:	Read-only

CL	Connection Time Limited

Determines whether a time limit has been set to the connection (value 1) or not (value 0). The CT attribute determines the connection's maximum duration.

Data type:	Integer
Value:	0 = no time limit 1 = time limit
Default value:	1
Suggested value:	A time limit is necessary on certain radio telephone lines. Limiting the connection time may be good practice also in other cases, if there is a risk that the connection is otherwise not broken.
Access:	No limitations

CT	Connection Time

The maximum time that a connection is allowed to last (in seconds). This attribute is significant only when time limiting is activated (CL = 1).

Data type:	Integer
Value:	0...600
Unit:	Seconds
Default value:	120
Access:	No limitations

CN	Connection

Dials the devices from the NET and breaks the telephone connections. A telephone call to a station or workplace is initiated by writing the telephone number to the CN attribute. The NET unit commands the autodialing modem to dial the telephone number. A successful dial is reported as a system message. To break the connection, write an empty string to the CN attribute.
When dialing a station, the station's link address is given at the end of the telephone number string. Type the letter S between the telephone number and the link address. This option is normally used to increase the communication performance on the multidrop lines.

Data type:	Text
Value:	Text string; maximum 25 characters
Default value:	Empty text string
Access:	No limitations

Example:

```
#SET NET1:SCN5 = "123456789S11"
```

CS	Connected Station

The station's link address is communicating with NET.

Data type:	Integer
Value:	0...65535 0 = autocaller is not defined or no communication
Default value:	0
Access:	Read-only

DD	Radio Disconnection Delay

Delay between the last data transfer and the line disconnection.

Data type:	Integer
Value:	0...32767

Table continues on next page

DD	Radio Disconnection Delay
Unit:	Seconds
Default value:	0
Access:	No limitations

MC	Modem Command

A modem can be controlled directly from SCIL with the AT (Hayes) commands. When an AT command is written to the MC attribute, the attribute is transmitted to the modem on a line. The MC attribute also reads the modem's response.

Data type:	Text
Value:	Text string, an AT/Hayes command
Default value:	0
Access:	No limitations

Example:

```
#SET NET1:SMC3 = ("AS0?")'
```

PU	Pulse Dialing

Determines which dialing principle is used.

Data type:	Integer
Value:	0 = tone dialing 1 = pulse dialing
Default value:	0
Access:	No limitations

RC	Remote Calls Enabled

States whether remote calls are enabled on a line. For example the NET unit can be called from the stations and connected to the line. This attribute applies to lines with autocaller (AC = 1).

Data type:	Integer
Value:	0 = remote calls not enabled 1 = remote calls enabled
Default value:	0
Access:	No limitations

RW	Radio Connection Wait Time

Normally, the DCD (Data Carrier Detect) signal indicates an active connection. There are cases, where this is not possible, for example, on the radiotelephone lines which use half-duplex links. The RW attribute defines the waiting time in these cases from the end of dialing until the transmission is started.

Data type:	Integer
Value:	0...32767
Unit:	Seconds
Default value:	0
Access:	No limitations

SR	Autocaller AT S Register

The autocallers' S registers follow the AT (Hayes) standard. Each autocaller that uses the AT command set has several S registers. The number of S registers in use and the meaning of the individual S register vary according to the autocaller model. Therefore, the contents of the S registers are not described in this manual. For more information on the S registers, refer to the manual of the modem.
When using the SR attribute, the following numbers of the S registers are accessed: 2, 6, 7, 8, 9, 10, 11 and 12. Other S registers can also be accessed by using the MC attribute. The S registers 11 and 12 cannot be set.

Data type:	Integer
Value:	See the modem manuals
Indexing:	Seconds
Access:	100 * line number + register number

Example:
The S register number 6 of line 2 in NET1 is set = 4:

```
#SET NET1:SSR206 = 4
```

4.4 Modbus PLC device configuration

The PLC station device is the heart of the SYS600 Modbus protocol converter. It converts communication messages from SYS600's internal protocol to the Modbus protocol and vice versa. The PLC device stores necessary information of protocol and address conversion in topic data. The PLC device also stores data which is scanned from an external device to an internal PC-NET database. The purpose of this storing is to minimize the amount of messages between the base system and PC-NET (PC-NET only sends changed data to the base system). Changed data is sent to the base system process database as RTU process data.

IU	In Use

Indicates whether the line is in use (value 1) or not in use (value 0).

Data type:	Integer
Value:	0 or 1
Default value:	0
Access:	No limitations

LI	Line Number

The number of the NET line the station is connected to.

Data type:	Integer
Value:	1...8
Access:	Read, conditional write

Setting this attribute is not needed when the station is created by using the DV attribute.

SA	Station Address

The station address of the Modbus master station (ID of the slave).

Data type:	Integer
Value:	0...255

Table continues on next page

SA	Station Address
Default value:	1
Access:	Read, conditional write.

When a serial Modbus is used, this value must be unique within the STA objects connected to the same line. If Modbus TCP is used, the same SA can be used many times within one line, if the IP-addresses are unique. STA objects connected to separate lines can always have the same value in the SA attribute.

AL	Allocation

Allocates the station to an application. When the AL attribute has the value 1, the station is reserved by the application specified by the AS attribute. All spontaneous messages from the station are sent to this application.

Data type:	Integer
Value:	0 or 1
Access:	No limitations

AS	Allocating Application

Specifies the allocating application of the station (see the AL attribute). The allocating application gets all the spontaneous process data from the station. This application is also the only one that is allowed to set the device communication attributes.

Data type:	Integer
Value:	0...250 0 = no application
Access:	Read-only

MI	Message Identification

Object address of the system messages.

Data type:	Integer
Value:	1...32760
Default value:	28000 + station number
Access:	Read, conditional write

MS	Message Application

The number of the application that is the receiver of the system messages generated by the line.

Data type:	Integer
Value:	1...250
Default value:	1
Access:	Read, conditional write

DC **Diagnostic Counters**

The values of the diagnostic counters which the NET unit keeps for the station. The counters have the following meaning:

1. Suspension information (0 = OK, 1 = suspended)
2. Suspension counter
3. Transmitted data messages
4. Transmitted command messages
5. Transmitted confirmation messages
6. Received data messages
7. Received command messages
8. Received confirmation messages
9. Received unknown messages

Data type:	Integer
Value:	1...65535
Index range:	1...20
Access:	Read-only

ML **Maximum Length**

Defines the maximum amount of words requested in each poll. This attribute can be tuned to get better performance and also if the Modbus slave device requires that certain blocks in its memory must be polled with a single poll.

Data type:	Integer
Value:	2..75
Default value:	31
Access:	Read, conditional write

RM **Running mode**

Consists of a set flags that control the behavior and the functionality of the Modbus master station. Each flag is one bit of this attribute.

Bit 0	If this bit is 0, the topics that are configured to have polling interval = 0 are not requested in any situation. This is the default mode. If this bit is 1, the topics or types Analog Value, Indication, Pulse Counter and Digital Value that are configured to have polling interval = 0 are requested in the communication start-up situation and when the GI command is issued.
Bit 1	If this bit is 0, the database update operation similar to communication start-up or the GI command is done automatically after a detected event buffer overflow situation in the device. This is the default mode. If this bit is 1, no automatic actions are made.
Data type:	Integer
Value:	0..65535, see above
Default value:	0
Access:	Read, conditional write

CE **Communication enabled**

Enables/disables the communication of the station object. Using the CE attribute, the SCIL application may disable communication without changing the status of the station object. In TCP/IP mode, the TCP connection to the IED is closed, if no other station object is communicating with the same IP-address. This attribute is useful, if the data reading is made irregularly or in a session-oriented manner.

Data type:	Integer
Value:	0 = communication disabled 1 = communication enabled

Table continues on next page

CE	Communication enabled
Default:	1
Access:	No limitations

Example:
#SET STA2:SCE=1 ;open a read session
#PAUSE 1
@AV_VECTOR = STA2:SAV(1..4) ;read analog values
@BI_VECTOR = STA2:SBV(16..31) ;read binary values
#SET STA2:SNP(3) = (1,0) ;demand scan for topic 3
#PAUSE 1
#SET STA2:SCE=0 ;close a read session

EA	Exception Address

Defines the information object address of an analog input process object which is used to indicate the reception of an exception to a poll message to the remote device. The value of the process object is the exception code received from the device. The meaning of each exception is listed in a table below.
If the EA attribute has a value of 0, no process object is updated. The default value is 0. If an exception is received to a SCIL command, the corresponding error code is returned, but the process object is not updated.

Value:	0..4095
Default:	0
Access:	Read/Write

Table 1: Modbus exception codes

Code	Description
1	ILLEGAL FUNCTION
2	LLEGAL DATA ADDRESS
3	ILLEGAL DATA VALUE
4	SLAVE DEVICE FAILURE
5	ACKNOWLEDGE
6	SLAVE DEVICE BUSY
8	MEMORY PARITY ERROR
10	GATEWAY PATH UNAVAILABLE
11	GATEWAY TARGET DEVICE FAILED TO RESPOND

4.4.1 An example configuration of a PLC device

The 8.4.3 (or newer) System Configuration Tool may be used for configuration since it supports the Modbus master protocol. The following figure shows an example configuration of a PLC device.

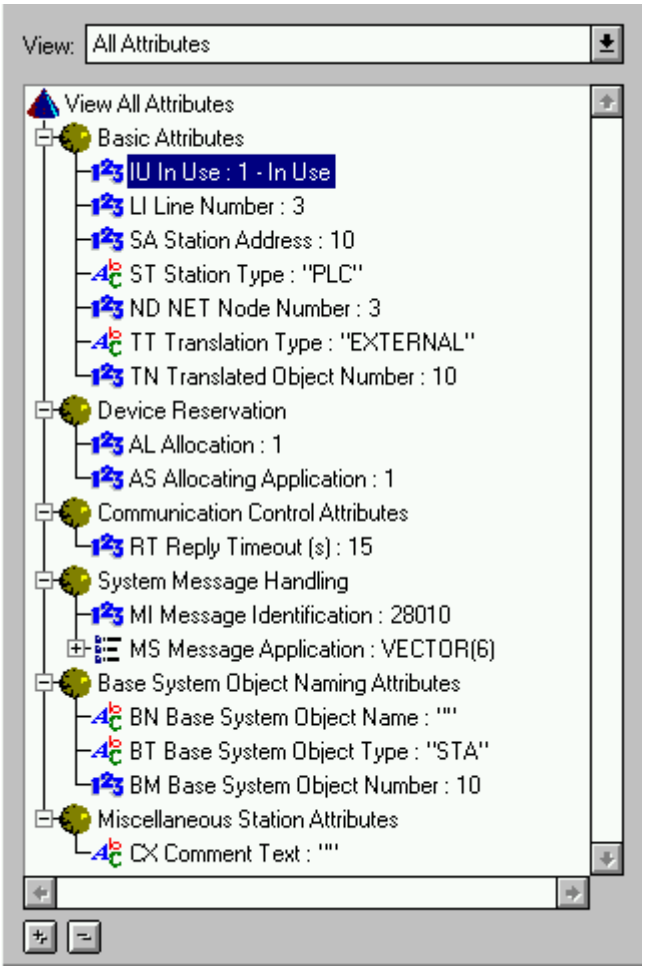


Figure 9: Example configuration of PLC device

The PLC device object is created using the Base System Configuration Picture, as described below.

1. Create a PLC station type:

   CREATE_OBJECT > STATION_TYPES >

STY NR:	28
TYPE NAME:	PLC
DATABASE TYPE:	RTU

2. Create a PLC Station object:

   CREATE_OBJECT > STATIONS >

STA NR:	1
TRANS.TYPE:	EXTERNAL
NODE NUMBER:	1
TRANSL. OBJ. NUM:	1
STATION TYPE:	PLC

4.4.2 Modbus TCP/IP attributes

IA **Internet Address**

The IP address or the host name of the remote host. The connection is established with a device in this address by using port number 502. The line must have been taken into use at least once before writing to this attribute.

Value: Any string, max 29 characters

Access: Read/write

This attribute accepts the IP address in the following form:

```
#SET STA1:SIA="10.0.0.1"
```

or as an alias name

```
#SET STA1:SIA="GRACE"
```

When an alias name is used, it must be defined in the TCP host file %windir \system32\drivers\etc\hosts.

Multiple station objects may have a same IA, this configuration may be necessary if the system contains, for example, Modbus/TCP vs Modbus serial converters.
IA-addresses may be entered with a colon:

```
STA1:SIA="10.0.0.1"
STA2:SIA="10.0.0.1:2"
STA3:SIA="10.0.0.1:3"
```

With this configuration a separate TCP session is created for each station object.
If the IA-addresses are entered without a colon:

```
STA1:SIA="10.0.0.1"
STA2:SIA="10.0.0.1"
STA3:SIA="10.0.0.1"
```

only one TCP session is created for the station objects having same IA. All communication takes place through this session. This is the most common configuration.
Same SA (Station Address) value can be used in multiple STA objects connected to the same line, if the IP addresses defined with IA are unique. This means that the default modbus address of the IED can be used as long as the IEDs are in separate IP addresses.
See the example in Section 5.3.4.
If the remote slave device uses a non-standard port for communication, it can be specified as follows:

```
SET STA1:SIA="10.0.0.1;503" ; remote device uses port 503
```

No space characters are allowed between the address and the port number. The port number must be in the range 1..65535.

CT **Connecting Timeout**

The maximum time of the TCP connect operation. The value of this attribute depends on the speed of LAN, remote station and the possible routers between SYS600 and the substation. The value should be lower than the HT attribute of the line, but high enough to enable reliable reconnection of the substation. In a multidrop configuration, a value that is too high may cause communication disturbances, if some of the stations are not available.

Value: 0..60000

Unit: Milliseconds

Default: 500 ms

Access: Read/Write

ET	rEconnecting Timeout

The interval of reconnecting attempt while communication is not established.

Value:	1..255
Unit:	Seconds
Default:	30
Access:	Read/Write

IS	Ignore Suspension

This attribute can be used to filter out occasional station suspensions caused by TCP connection breaks and server disconnections. Suspensions caused by response timeouts are not ignored. The attribute value represents the amount of suspensions that are filtered before a station is set to suspended state. The value of IS applies to all station objects that have the same Internet Address (IA) attribute value and are connected to the same line. When a connection is lost, the time when the master station is set to suspended state can be calculate with the following formula

Suspension time = Header Timeout (HT) * Enquiry Limit (EN) * 2 + (rEconnecting Timeout (ET) * Ignore Suspension (IS))

The ET value should be bigger than the part “HT * EN” in the formula.

With the default IS value (0) the stations are set to suspended state every time the TCP connection is detected to be broken. The IS values bigger than 0 should be used only in configurations where each station object in a line has a unique Internet Address in IA attribute.

Attribute has an effect only in TCP mode.

The IS attribute should only be used if unnecessary suspensions are constantly present. Otherwise it is recommended to extract the reason for these suspensions instead of using the IS attribute values bigger than 0.

Data type:	Integer
Value:	0..30
Default:	0
Access:	Read/Write

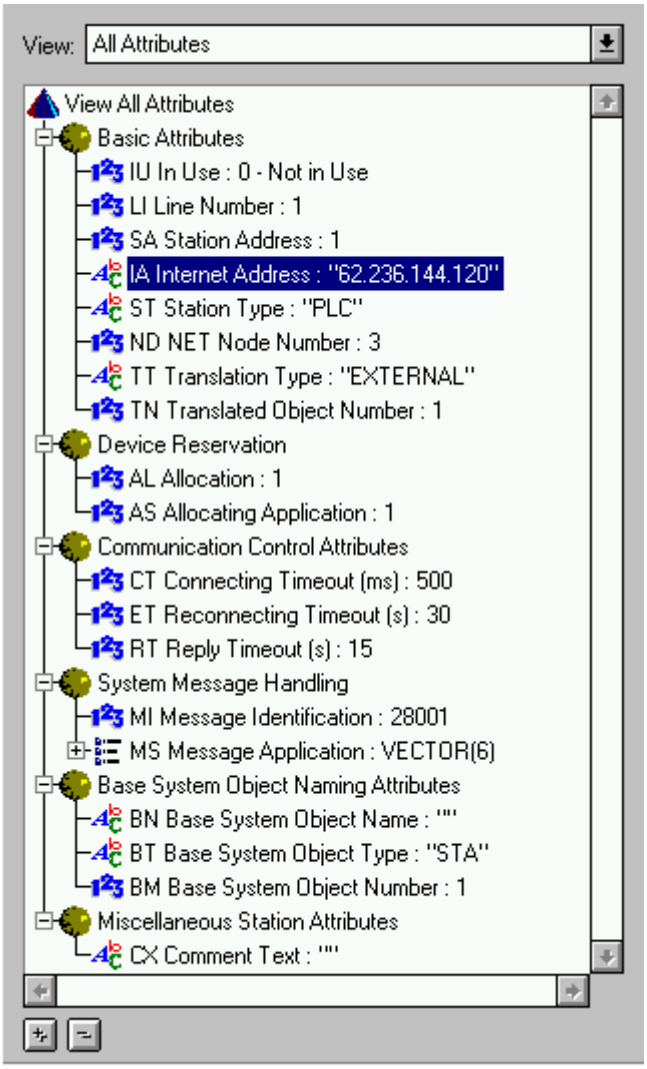


Figure 10: Example configuration of PLC device

4.5 Topics of a PLC device

The scanning of external device data is controlled by the PLC device topics. Since the PLC device can contain several different topics (max. 100), it is possible to divide the memory and IO (input/output) into separate external device areas. These areas can contain different types of data and they can have individual scanning intervals.

4.5.1 Read topic length limitations

4.5.2 Read messages

When creating a read topic in Modbus master (types 6, 7, 8 or 9) and the topic interval is >0, i.e. it is polled, the recommended limit for the requested data length is 30 registers (30 items of 16-bit type or 15 items of 32-bit type or 480 single indications). If the topic interval is 0 and the values are read using station attributes AV, DV and BV, the length of the topic may be bigger. See the corresponding attribute descriptions for maximum messages lengths.

4.5.3 Write messages

When using process objects of output type, only a single value can be written. If there is a need to write multiple values at the same time, a write operation using the station attributes AV, DV and BV must be done. The maximum vector length of the write operation is described in the corresponding attribute descriptions. The configured topic must be long enough to cover all values written. The Modbus slave devices may also limit the amount of registers written in one request.

4.5.4 Topic parameters

The topic parameters are stored in the PLC station topic parameter table. The memory needed by the topic is dynamically reserved based on the address and format parameters. The parameter table is filled in by writing a vector to the TP attribute, which is also possible to read.

Syntax of a topic configuration command for non-analog topics is as follows:

```
STA'n':STP'index'=(Allocation, FirstObjectAddress, LastObjectAddress,
Type, BaseAddress, Format, Interval, delta, [Bitcount])
```

Syntax of a topic configuration command for analog value topics is as follows:

```
STA'n':STP'index'=(Allocation, FirstObjectAddress, LastObjectAddress,
Type, BaseAddress, Format, Interval, delta, [NominalValue],[Percentage])
```

For analog value topics, parameters Delta and parameters NominalValue and Percentage provide alternative ways to decrease the data updating messages from NET to base system. In case parameters NominalValue and Percentage are defined and they both are non-zero, the filtering described in chapter 'NominalValue and Percentage' is used. In this case, the delta parameter should be set to 0.

The 8.4.3 (or newer) System Configuration Tool may be used for configuration, since it supports the Modbus master protocol. The following figure is an example of topic configuration under the PLC station type.

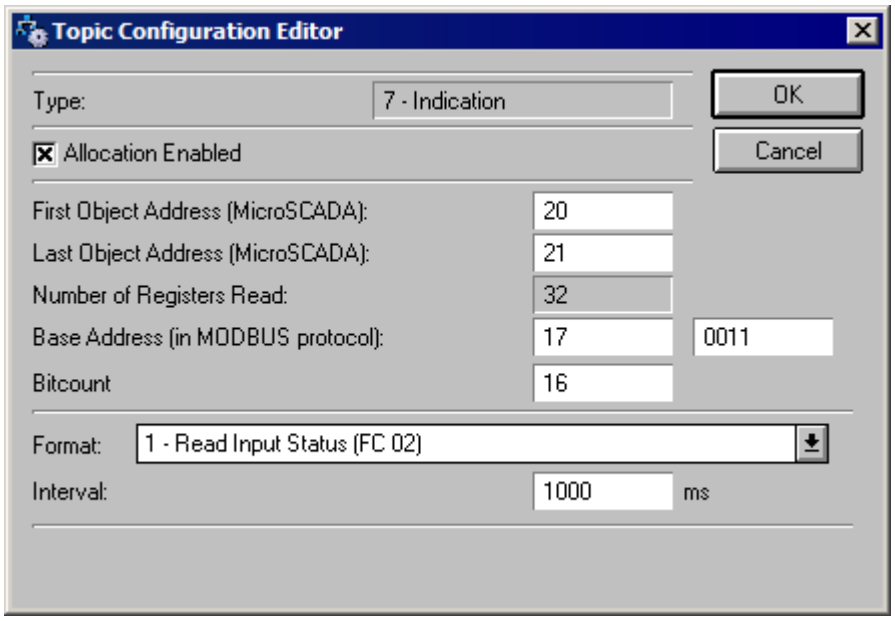


Figure 11: Topic configuration with System Configuration Tool

The topic configuration vector contains the following parameters:

4.5.4.1 Allocation

This item specifies whether the topic is in use or not. The memory needed for the topic is reserved, when the topic is taken into use.

4.5.4.2 FirstObjectAddress

This parameter specifies the First SYS600 Process Object Address used with this topic. The First Object Address and the Last Object Address can be chosen freely. The object address and object type parameters together specify the actual process object address (OA), where the first item in the topic is stored. See below:

OA = 4096 * Object_Type + FirstObjectAddress

This calculation is also done automatically in the process object tool. The user has to add the same address parameters as with the topic. When the actual object address is read, the calculated object address is shown instead of the parameter value.

If the topic type is SPA_EVENT (101), the given value is the address of the bitstream process object which is updated with the contents of the event table (RCE file). If the topic type is REF 542plus time synchronization (102), the given value is not used.

4.5.4.3 LastObjectAddress

This parameter is the object address of the last topic item. The First Object Address and Last Object Address define how many registers/items can be read/written with the topic. These items are also referred to as block addresses when talking about binary indications. The number of items handled by the topic is calculated as shown below:

For topic types Single or Double Indication

Number of items = (Last Object Address – First Object Address)*16+BitCount

For topic types Object Command, Digital Setpoint, Analog Setpoint, Analog Value, Pulse Counter and Digital Value:

Number of items = LastObjectAddress - FirstObjectAddress + 1

If the topic type is SPA_EVENT (101) or REF 542plus time synchronization (102), the given value is not used.

4.5.4.4 Type

This parameter specifies the data type of the process objects. The following table shows the possible data types of a PLC device.

Table 2: Possible data types of a PLC device

Type	Type of process object	Possible data formats
1:	Object command	IO_BIT, M_BIT
3:	Digital set-point	INT, WORD
4:	Analog set-point	CHAR, INT, WORD, LONG, FLOAT
6:	Analog value	CHAR, INT, WORD, LONG, FLOAT
7:	Indication (single or double)	IO_BIT, M_BIT
8:	Pulse counter	LONG
Table continues on next page		

Type	Type of process object	Possible data formats
9:	Digital value	INT, WORD
101:	SPA_EVENT for REF 542plus	Format not used
102:	Time sync for REF 542plus	Format not used

With the process object type "indication", one object address (OA) contains 16 bits.

4.5.4.5 Base Address

This Modbus address is the first item address of the topic in the PLC memory. With binary indications the address space is 1 - 65536. With 16 bits registers it is 1 - 4096. If the topic type is Single or Double Indication, the value referred to by the Base Address in the memory area of the IED is stored to a process object that has Block Number = First Object Address and Object Bit Address = 0. For topic types Object Command, Digital Setpoint, Analog Setpoint, Analog Value, Pulse Counter and Digital Value, the block address of the process object corresponding to a given BaseAddress is FirstObjectAddress. The block addresses of the process objects should be between FirstObjectAddress and LastObjectAddress.

If the topic type is SPA_EVENT (101), the given value refers to the event table number in the REF 542plus relay and must have some of the following values:

Value	Description
2:	RCE Table (REF 542plus historical register of events)
5:	RCE extended Table (REF 542plus historical register of events in extended form)
11:	(RCE table): as File 2 - Read and Clear of Events
12:	(RCE extended table): as File 5 - Read and Clear of Extended Events

If the topic type is REF 542plus time synchronization (102), the given value must be 7 (Time and Date).

4.5.4.6 Format

This parameter specifies how data is stored in external devices. Possible formats are shown in the following table.

Table 3: Possible save formats for external devices

Format	Code	Description
IO_BIT	1	Bit in PLC's input or output.
M_BIT	2	Memory bit in PLC's working memory.
CHAR	3	Unsigned 8 type object in PLC's registers. A register can allocate two CHAR.
INT	4	One register in PLC's memory. The MSB bit is used as a sign bit.
WORD	5	One register in PLC's memory. The object is used in an unsigned form.
LONG	6	Signed 32-bit object, which needs two registers from PLC's memory in msw-lsw order.[1)]
MSB_LONG	7	Signed 32-bit object, which needs two registers from PLC's memory in lsw-msw order.
F32_TYPE	8	Floating point type which needs two registers from PLC's memory.

Table continues on next page

Format	Code	Description
MSB_F32_TYPE	9	Floating point type which allocates two registers in PLC's memory. The registers are in reversed order compared to format 8.
IN_WORD	10	Input register of PLC (3x references).
IN_CHAR	11	Unsigned 8 type object PLC's input registers. A register can allocate two IN_CHARs.
IN_INT	12	Input register of PLC. The MSB bit is used as a sign bit.
IN_LONG	13	Signed 32-bit object which needs two input registers from PLC in msw-lsw order.
IN_FLOAT	14	Floating point type which needs two input registers from PLC.
IN_MSB_LONG	15	Signed 32-bit object which needs two input registers from PLC in lsw-msw order.
IN_MSB_FLOAT	16	Floating point type which needs two input registers from PLC. The registers are in reversed order compared to format 14.
MSB_INT	17	One register in PLC's memory, the bytes are in reversed order. The MSB bit of the first byte of the register is used as a sign bit.
MSB_WORD	18	One register in PLC's memory, the bytes are in reversed order. The object is used in an unsigned form.
IN_MSB_INT	19	Input register of PLC, bytes are in reversed order. The MSB bit of the first byte of the register is used as a sign bit.
IN_MSB_WORD	20	Input register of PLC (3x references), the bytes are in reversed order. The object is used in an unsigned form.

1) The most significant word - the least significant word (and vice versa)

If the topic type is SPA_EVENT (101) or REF 542plus time synchronization (102), the given format value is not used.

If the topic type is Analog set-point (4) and bit 6 of the given format value is set, analog set-point commands sent will always use Function 16, Write Multiple Registers. If bit 6 is not set in the given format value, Function 6, Write Single Register, is used when only register is written. This is the default functionality. Formats WORD, MSB_WORD, IN_WORD and IN_MSB_WORD can be used with type 7 = Indication. With this setting, the binary and double binary input process objects can be updated with the values requested using modbus functions 03 Read Holding registers (WORD and MSB_WORD) and 04 Read Input register (IN_WORD and IN_MSB_WORD).

4.5.4.7 Interval

This is the frequency with which topic data is read from an external device. The interval units are milliseconds. If the interval is 0, the topic is not polled cyclically. The maximum polling interval is 2147483647 milliseconds (> 24 days). See the description of the attribute Table 5 for on-line changing of the polling interval. If bit 0 of RM attribute is set, topics with interval = 0 have a special handling.

4.5.4.8 Delta

If the topic type is an analog value (type=6), the delta value is used to minimize the amount of updating messages from the NET to the base system. The new analog value is sent to the base system, when the change or the sum (integral) of changes is bigger than the delta value. If the format of the topic is 8,9,14 or 16 (float), delta cannot be used but NominalValue and Percentage described below must be used instead.

If the topic type is SPA_EVENT (101), the given value is not ignored.

If the topic type is REF 542plus time synchronization (102), the given value is a constant correction offset in milliseconds. A non-zero value can be used, if the communication hardware causes a predictable delay in message transmission.

An alternative way to reduce the amount of updating messages from the NET to the base system is the usage of NominalValue and Percentage parameters below.

Delta and NominalValue / Percentage parameters cannot be used together in the same topic. If NominalValue / Percentage is needed, delta parameters values MUST be set to 0. If Delta is needed, NominalValue / Percentage MUST be set to 0.

4.5.4.9 Bitcount

This parameter is meaningful only when the topic type is indication. The bitcount is an optional parameter which defines the count of the requested indication bits in the input word referred to with the parameter Last Object Address. The value range of the bitcount parameter is 1..16. If the bitcount parameter is not defined, value 16 is used as default. The bitcount value 16 means that each bit from every block, from the First Object Address to the Last Object Address are requested. If the bitcount value is 2, 16 bits are requested from each block, except only two bits (LSB) from the block set as the Last Object Address.

When using double indication process objects, the bitcount should always be an even number.

The last indication read from the address = BaseAddress+(LastObjectAddress-FirstObjectAddress)*16+Bitcount

Example of creating binary topics

If binary indication registers from 1 to 32 and from 49 to 51 need to be read, it can be done with one or two single indication topics. With one topic also registers between 32 and 49 are requested and the topic parameters are:

First Object Address = 1

Last Object Address = 4

Base Address = 1

Bitcount = 3

With this definition, registers 33..48 are uselessly transmitted from the IED, but on the other hand, the required indications are read from the slave device at the same time.

With two topics, the first topic reads the bits from 1 to 32 and the second reads only the three bits from 49 and 51. The first topic's parameters are:

First Object Address = 10 (The addresses could be also 1 and 2, or other arbitrary values)

Last Object Address = 11

Base Address = 1

Bitcount = 16

and the second topic's parameters are:

First Object Address = 12

Last Object Address = 12

Base Address = 49

Bitcount = 3

Dividing the memory area into two topics usually makes sense, if the amount of unnecessary items between the necessary items is big, or if there is a reason to have different polling intervals for different areas.

If double indication process objects are used, the process objects are updated only if both bits are defined in the topic. The process object values in this case are:

lower indication = 0, higher indication = 0 -> DB value = 0

lower indication = 1, higher indication = 0 -> DB value = 1

lower indication = 0, higher indication = 1 -> DB value = 2

lower indication = 1, higher indication = 1 -> DB value = 3

4.5.4.10 NominalValue and Percentage

An alternative for the Delta parameter are parameters NominalValue and Percentage to reduce the amount of process object updating messages from the NET to the base system. This setup may be useful in big systems where continuous analog updates cause useless load to system servers or control centers. If these parameters are set, a dynamic deadband functionality is used in which the deadband value varies depending on the given NominalValue, Percentage and the received value.

The dynamic calculation of the deeadband value guarantees that small changes in value still update the process objects although the received values are remarkably smaller than the given nominal value. For this functionality both NominalValue and Percentage parameters need to be set and the Delta parameter of the topic must be set to value 0.

If the NominalValue and Percentage are given, a base deadband value is calculated from the given values with formula

```
BASE_DEADBAND = NOMINALVALUE * PERCENTAGE / 100
```

For example with NominalValue 6000 and Percentage 0.5% the base deadband value is (6000 * 0.5 / 100) = 30.

For all received analog values that are larger than the NominalValue the deadband is always equal to the base deadband value:

```
USED_DEADBAND = BASE_DEADBAND
```

For analog values smaller than the NominalValue a scaled deadband is calculated with the following formula:

```
USED_DEADBAND = BASE_DEADBAND - ( ( BASE_DEADBAND / ( 1 +
( CURRENT_VALUE * PERCENTAGE / 100 ) ) - 1 )
```

If a new analog value differs from the last stored value more than the USED_DEADBAND value, an update is sent to the process object in the base system and the received value is stored for the next comparison.

The NominalValue range is 0 – 2147483647 and the Percentage value range is 0– 100%.

The filtering is necessary especially when the values are reported to the network control center using the COM500*i* application or when the PC-NET is running in a different computer than the database receiving the data. If reported without filtering and there are plenty of analog updates from the IEDs, small analog changes may cause useless load in the control center.

Deadband values are calculated using floating point arithmetics, which means that rounding errors are possible when calculating if the value should update or not. Rounding errors are present also when the received values are integer values.

Example:

NominalValue is 6000 and the change Percentage is 0.5% and thus the base deadband value is +-30. Using the formula the used deadband value is calculated for values smaller than the NominalValue:

For updated value 10000 the next used deadband = 30 (is bigger than NominalValue)

For updated value 1000 the next used deadband = 30 - ((30 / (1 + (1000 * 0.5/100)) - 1) = 26

For updated value 100 the next used deadband = 30 - ((30 / (1 + (100 * 0.5/100)) - 1) = 11

For updated value 10 the next used deadband = 30 - ((30 / (1 + (10 * 0.5/100)) - 1) = 2

If the last updated value was 100 and value 110 is received, the process object is not updated.

If the last updated value was 100 and value 114 is received, the process object is updated.

If the last updated value was 1000 and value 1025 is received, the process object is not updated.

If the last updated value was 10000 and value 9960 is received, the process object is updated.

4.5.4.11 An example configuration of data topics

The following example shows how to configure data topics of a PLC device.

```
#SET STA1:SIU=0
    #SET NET1:SIU4=0

; INIDICATION BLOCK
; OA 28673..28673, IO_BIT, SINGLE INDICATION, BASE ADDRESS 1
#SET STA1:STP(1)=( 1,1,1,7,000,1,1000,0)

;DIGITAL VALUE BLOCK
; OA 36865..36866, WORD_TYPE, DIGITAL VALUE, BASE ADDRESS 400
#SET STA1:STP(2)=(1,1,2,9,400,4,1000,000)

;DIGITAL VALUE BLOCK
; OA 36867..36868, INPUT REGISTER TYPE, DIGITAL VALUE, BASE 400
#SET STA1:STP(2)=(1,3,4,9,400,10,1000,000)

;ANALOG VALUE BLOCK
; OA 24576..24577, INT_TYPE, ANALOG VALUE, BASE ADDRESS 200
#SET STA1:STP(3)=(1,1,2,6,200,4,10000,10)

;ANALOG VALUE BLOCK
; OA 24578..24578, LONG TYPE, ANALOG VALUE, BASE ADDRESS 400
#SET STA1:STP(4)=(1,3,3,6,400,6,1000,00)

;ANALOG VALUE BLOCK
; OA 24579..24579, LONG TYPE, ANALOG VALUE, BASE ADDRESS 200
; NOMINALVALUE 6000, PERCENTAGE 0,5%
#SET STA1:STP(5)=(1,4,5,6,200,4,2000,0,6000,0.5)

;OBJCET COMMAND BLOCK
; OA 4097..4126, IO_BIT, OBJECT COMMAND, BASE ADDRESS 1
#SET STA1:STP(6)=(1,1,30,1,0,1,0,0)

;DIGITAL SETPOINT
; OA 12288..12288, WORD TYPE, DIGITAL SETPOINT, BASE ADDRESS 254
#SET STA1:STP(7)=(1,1,3,3,254,5,0,0)
```

```
;ANALOG SETPOINT
; OA 16385..16387, INT TYPE, ANALOG SETPOINT, BASE ADDRESS 314
#SET STA1:STP(8)=(1,1,3,4,314,4,0,0)

;ANALOG SETPOINT
; OA 16388..16390, INT TYPE, ANALOG SETPOINT, BASE ADDRESS 320, FUNC16
ALWAYS USED
#SET STA1:STP(9)=(1,4,7,4,320,4+64,0,0)

#SET NET1:SIU4=1
#SET STA1:SIU=1
```

Example of a topic using function 04:

```
; OA 24578..24578, IN_LONG TYPE, ANALOG VALUE, BASE ADDRESS 400 #SET
STA'STA':STP(1)=(1,3,3,6,399,13,1000,0 )
```

Examples of a topic using the Bitcount parameter:

```
; INDICATION BLOCK
; OA 28673..28673, IO_BIT, SINGLE INDICATION, BASE ADDRESS 1 (12 bits
requested)
#SET STA1:STP(1)=( 1,1,1,7,0,1,1000,0, 12)

; INDICATION BLOCK
; OA 28673..28674, IO_BIT, SINGLE INDICATION, BASE ADDRESS 1 (28 bits
requested)
#SET STA1:STP(1)=( 1,1,2,7,0,1,1000,0, 12)
```

4.5.5 Address conversion between RTU and Modbus

4.5.5.1 Addressing system of Modbus protocol

The Modbus protocol and Modicon PLCs divide addressable memory into four different areas as shown below.

Address:	Description:
0xxxx	Discrete outputs and discrete coils
1xxxx	Discrete inputs
3xxxx	Input registers
4xxxx	Holding registers

The external communication unit refers to these address areas by using different message functions as shown below.

Function:	Description:
01	Read coil status (0xxxx)
02	Read input status (1xxxx)
03	Read holding registers (4xxxx)
04	Read input registers (3xxxx)
05	Force single coil (0xxxx)
06	Write single register (4xxxx)
15	Force multiple coils (0xxxx)
16	Write multiple registers (4xxxx)

Many third party Modbus protocol converters (for example Siemens CP525) do not separate input and memory areas, which means that functions 01 and 02, as well as 03 and 04 point to the same address area.

A REF 542plus related function uses the following Modbus function codes:

20	Read General Reference
21	Write General Reference

See Section Section 4.5.5.3 and Section 4.5.5.3 for more information about the REF 542plus related functions.

4.5.5.2 Addressing systems of RTU process objects in SYS600

These rules apply to topic types 1...9 only. For topic type SPA_EVENT (101), see Section 4.5.6.

The object address (OA) of the RTU type process objects are packed 16 bit values, which contain the actual address and also the type of data as shown below:

OA = 4096*Object_Type + Object_Address

Object_Type is one of the object types described in the table below.

Table 4: Object type codes and their process object types

Object_Type Code	Process object type
1:	Object command
3:	Digital set-point
4:	Analog set-point
6:	Analog value
7:	Indication (single or double)
8:	Pulse counter
9:	Digital value
101:	SPA Event
102:	REF 542plus Time Synchronization

4.5.5.3 Addressing Modbus objects from SYS600

This section describes the addressing of Modbus objects from SYS600. The following table shows the relationship between the Modbus message functions and topic parameters.

Table 5: Modbus message functions and topic parameters

Function	Topic type	Topic format
01	7	M_BIT
02	7	IO_BIT
03	6, 8, 9	CHAR, WORD, INT, LONG, FLOAT
04	6, 9	IN_WORD, IN_CHAR, IN_INT, IN_LONG, IN_FLOAT, IN_MSB_LONG, IN_MSB_FLOAT
05	1	IO BIT or M BIT
Table continues on next page		

Function	Topic type	Topic format
06	3, 4	CHAR, WORD, LONG, FLOAT
15	1	M BIT (only when writing to the DI attribute
16	3, 4	CHAR, WORD, LONG, FLOAT (when vector written)

Topic type = Process object type

4.5.6 REF 542plus event reading and time synchronization

The topic type SPA_EVENT 101 is used to read the event table contents from the REF 542plus relay. The events are stored in four alternative tables, which are RCE files. The selected event table is defined using the Base Address parameter of the topic. No more than one topic of type 101 can be configured to each STA object.

If the topic of type 101 is configured, the number of available events is continuously polled with the interval given with the topic parameter Interval. If the event count is bigger than zero, the event table reading sequence starts and the contents of the event table is transmitted to a bitstream process object with OA= FirstObjectAddress. The events are read in blocks of max. 8 events.

If the configured event table number is 2 or 5 (no clear operation made by the REF 542plus relay), an automatic event clearing command is sent to the relay when the events have been transmitted successfully to the bitstream process object. A SCIL application is needed to process the contents of the bitstream process object.

If an event buffer overflow situation occurs in the REF 542plus device, the device status point is updated with status PLCP_EVENT_BUFFER_OVERFLOW = 13829 and a GI is issued automatically, if it is enabled with the RM attribute bit 1. The process database is kept in a valid state in the overflow situation.

The topic type REF 542plus time synchronization 102 is used to send the time synchronization to a REF 542plus relay. The synchronization is made cyclically if the Interval parameter of the topic is bigger than 0. If the Interval parameter is 0, the synchronization can be made from the SCIL application using the station attribute SY. Synchronization is needed in order to have the correct timestamps for the events.

4.6 Communication adjustment guidelines

MicroSCADA has multiple adjustable attributes that affect the communication. The default values might not be suitable for each system and can sometimes cause communication problems. Here are listed the steps of what should be done when the communication is not working correctly. Note that these advices are for serial line only.

4.6.1 Basic checks

In case of a communication does not start properly, following checks are worth to do before any further investigation:

- Check that the serial port defined with line attribute SD matches with the computes serial port. Check also that no error is reported to the notify window when the system or PC_NET is started.
- Check that the stations addresses match in master and slave.
- Check that the baud rates and other Data Transmission Attributes (in MicroSCADA System Configuration Tool) match in master and slave.
- Check the length limitations for analog value requests and confirm that the analog topic parameters are within those limitations.

4.6.2 Message sending

If the remote end does not receive the messages sent from the line in question, following hints may help:

- Check the FIFO buffers of the used serial port. If RTS signal is used to control the data carrier, communication may function better if the FIFO buffers are disabled (or set to 1).
- Check the RTS Keepup Delay (RY) attribute value. If the RTS signal is used to control the data carrier and it is reset too early, the end of the message might be left out when the message is sent. With RY attribute the closing of the RTS signal can be delayed. On the other hand, if the RY delay is too long, the remote device may start transmission when the data carrier is not available. This causes an error.
- Check the COM Port Mode (CM) attribute. With the bit 3 setting the length of the message is used to calculate the RTS keep up time. See attribute documentation for more information.

4.6.3 Message reception

If the remote end send message but those are not received correctly by the line in question, following hints may help:

- Carrier Block (CB) attribute might be blocking the incoming messages. If the CB attribute is 0, carrier blocking is not used and messages are received regardless of the DCD state. If the CB attribute is 1, carrier blocking is used and the DCD must be signaled while characters belonging to the message is received.
- Check the DE attribute values. The DE attribute delays the sending of the message after the RTS signal is raised.
- Check that the Header Timeout (HT) and Response Timeout (TI) attributes lengths are long enough for the communication. The TI time could be too short in cases when the response is long and the baud rate small.
- If the messages are sent and received correctly but are not processed, check CRC or other errors from the Diagnostic Counters in System Configuration Tools online-mode.
- The usage of the protocol analyzer and the bit 1 or line attribute AU (Analyzer usage) will help in the tuning of the attributes DE, HT, TI, RI and RY. If bit 2 of the attribute AU is set, internal information related to completion of the write operation to the serial driver is display in the analyzer output.

4.6.4 Data updating

If the remote end sends messages and it seems like those are received correctly by the line in question but the process objects are not updating, following hints may help:

- Check that the attached process objects addresses are correct.
- Check that the requested process objects addresses are correct in the topics.
- Check that the STA object is allocated to the MicroSCADA application in which the process objects are located. Related STA object attributes are AL (Allocation) and AS (Allocating application).
- If the base system attribute TN (Translated Number) or ND (Node number) for the STA object has been modified by the SCIL application, check that those refer to correct STA object in correct PC_NET node

Section 5 Using the Modbus Master Protocol

This chapter describes how to transfer data between SYS600 and an external device by using the Modbus protocol serial line or TCP network.

5.1 Requirements

The following software is required:

- SYS600
- Operating system - Windows

General knowledge about the Modbus protocols and PLC programming is also a requirement.

Install the software as described in their respective manuals. The installation of SYS600 is described in SYS600 Installation and Administration Manual.

5.2 Accessing Modbus data using process objects

The most straightforward way to read data from a PLC device is to use the tools in the SYS600 package. The first step is to create topics to PC-NET, for example in System Configuration Tool and then request data from the device. Creating topics is explained in Section 4.5.4

Below is an example of a topic for indication data:

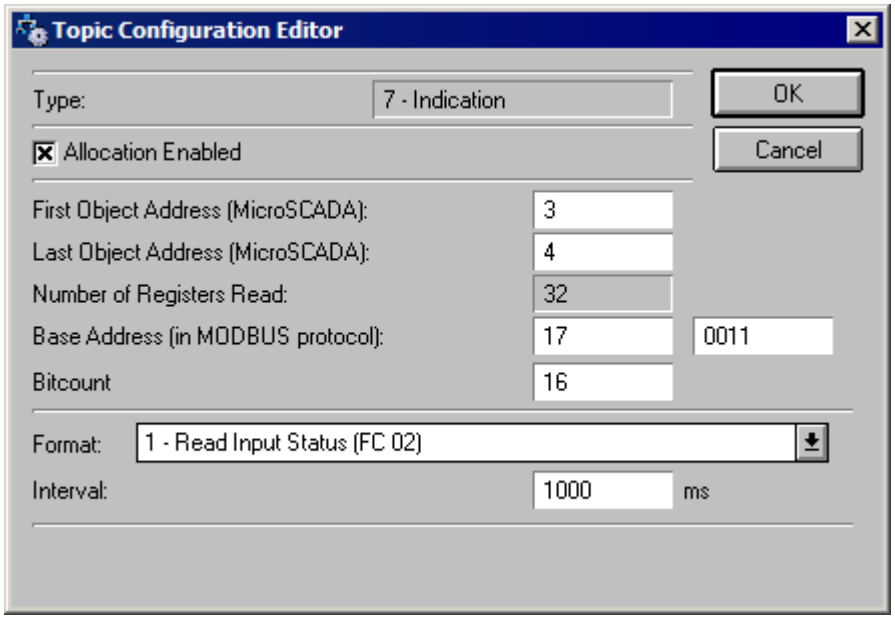


Figure 12: Example of topic configuration

The requested data can be used by the SYS600 database, if the RTU type process objects are created. The Object Navigator Tool can be used to create these objects. To be more specific, the process objects' type needs to be RTU-200 for the Modbus master. The following example shows a process object suitable for the topic above.

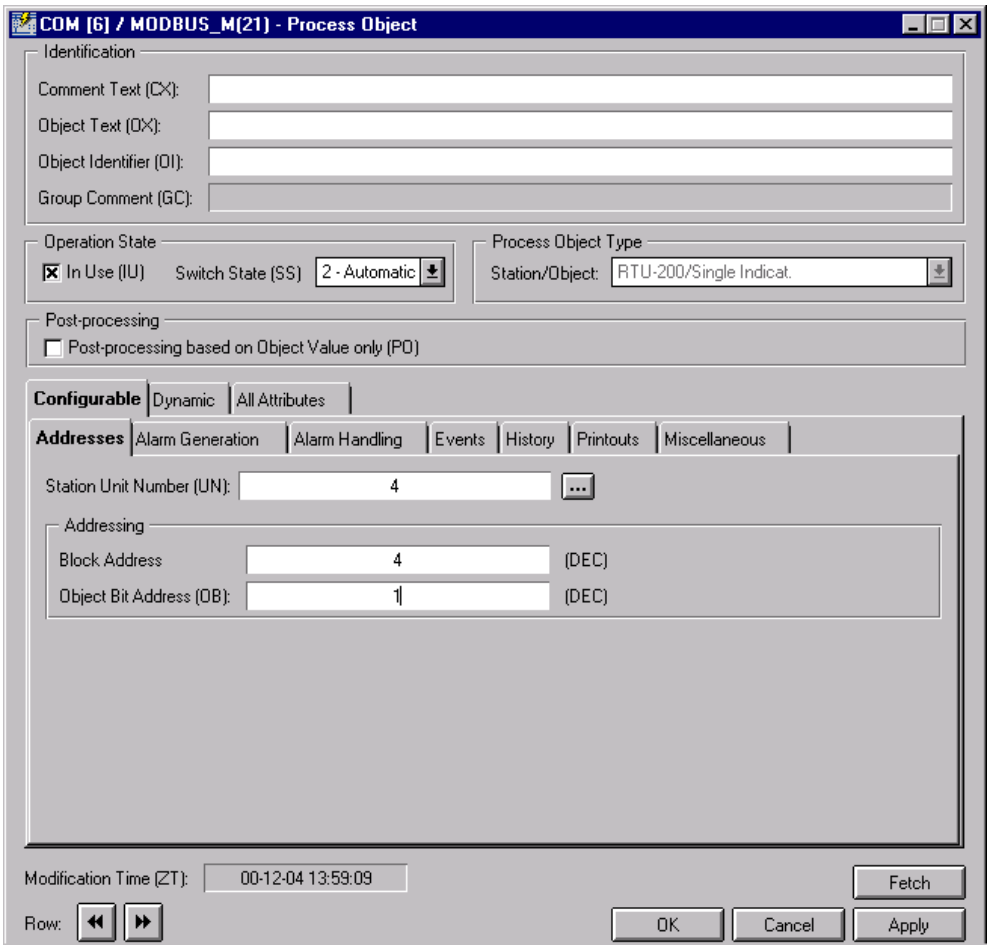


Figure 13: Example of process object

The topic requests data in the Modbus address space from bit address 17 to 48 (2*16). One SYS600 block object address contains 16 bits. The block address needs to be a value between the First and the Last Object Address defined in the topic. The single indication process object in the example above uses the bit 1 of block address 4 and the corresponding Modbus address of that bit is 17+(4-3)*16+1=34. For the rest of the binary data in the same topic, a process object must be created for every bit. With double binary, each process object uses two consecutive bits and the process objects' type is the double binary RTU object. While double binary uses 2 bits, the values stored can be in the range 0..3.

Analog format data handling is similar, except that no bit handling is used. The process object used refers to the defined Modbus register address. For instance, the topic for analog input is defined in the object addresses 100-101 and the Modbus base address is 10. In this case, the process object with the address 100 shows data from the Modbus register address 10 and a process object with the address 101 shows data from the register address 11.

5.3 Configuring the device attribute interface

Another way to read data from a PLC device is to use the NET attribute interface. A PLC station contains its own database based on the created topics. The database can be accessed via the attributes described below.

5.3.1 Attribute interface of a PLC device

The attribute interface of a PLC device is described in the following.

5.3.1.1 Communication attributes

The base system communicates with the PLC by using communication attributes. If a communication attribute is accessed from SCIL, the information stored to topics created for the station objects is used to form a modbus request. When the SCIL application writes or reads the communication attribute, the data is written or read to/from the process device as described in the attribute descriptions below. Data read or written using the attributes BV, SI, DI, AV or DV is not stored in the database in NET. It is sent to the process device or returned to the SCIL application as such. With attributes BV, SI, DI, AV or DV, the given indices of the attribute must match some of the created topics with the corresponding topic type. The formulas for the index calculations are described in the attribute descriptions below. The index must be between IndexFirst and IndexLast.

The communication attributes are as following:

BV **Bit Value**

This attribute refers to a vector of bits in the PLC memory or IO. When this attribute is written, the type of the topic must be 1 = Object Command, Writing to this attribute causes the transmission of command function 15 (Force multiple coils) to the Modbus line, even when there is only one item.

When this attribute is read, the type of the topic must be 7 = Indication. Reading this attribute returns a single indication value or a vector of single indications.
With format = 1 (IO_BIT), the used Modbus function in the request is 02 (Read input status) and with format = 2 (M_Bit), the function is 01 (Read coil status).

The indices for BV attribute are calculated as follows:

index = Object_Address * 16 + bit number (when reading from the BV attribute)
index = Object_Address (when writing to the BV attribute)

First index and last index define the range of bit values to be read or written. The starting addresses in PLC are calculated as follows:

PLC address = ((index/16) - FirstObjectAddress)*16+(index modulo 16) + Base Address (when reading from the BV attribute)
PLC address = (index - FirstObjectAddress)+Base Address (when writing to the BV attribute)

Access: R/W

Index range: 0..65535

Value range 0..1

Example of reading

```
@RET=STA1:SBV(80..95)
; returns a vector of 16 single indications
; If e.g the topic contains settings FirstObjectAddress = 4,
BaseAddress=300, the PLC addresses 316..331 are read
```

Example of writing

```
#SET STA1:SBV(16..19)=(1,1,1,1)
; issues a 'Force multiple coils' command to addresses 16..19
; If e.g. the topic contains settings FirstObjectAddress = 16,
BaseAddress=1000, the PLC addresses 1000..1003 are written
```

SI **Single Indication**

This attribute refers to a single bit in the PLC memory or IO. Writing to this attribute changes the status of one bit in the PLC. The SI attribute can only be used with topics whose type is bit. The index for the SI attribute is calculated as follows:
index = Object_Address * 16 + bit number (when reading from the SI attribute)
index = Object_Address (when writing to the SI attribute)

Access: R/W

Index range: 0...65535

Value range: 0...1

DI **Double Indication**

This attribute also refers to the PLC bit memory or bit IO. Double indication has (as single indication) two logical states (0 and 1), but in the double indication both directions have their own bits (0 -> 01, 1->10). The DI attribute can only be used with topics whose type is bit. The index for the DI attribute is calculated as follows:
index = Object_Address * 16 + bit number, when reading from the DI attribute
index = Object_Address, when writing to the DI attribute
The use of Double Indication needs support from the PLC application program, because the PLC program language does not directly support double indication data type.

Access: R/W

Index range: 0..65535

Value range: 0... 3

AV **Analog Value**

This attribute refers to the register data in the PLC memory. The register is always a 16-bit word in the PLC memory, but the PLC program can use registers in 1, 2 or 4 byte format. The PLC program can also use successive registers in different formats. If the format (=data type) of the topic is not the same as in PLC, the data type conversion must be made in the MicroSCADA application. If the data is requested using only the AV-attribute, the topic interval must be set to 0.

If AV is read, the type of the topic must be 6 (analog value) and the returned value is a vector of values read from the registers of the PLC. The meaning of the indices are:

First index = First object address
Last index = Last object address

First and last index of the AV reading do not need to be the same as the defined object addresses for the topic.
Returned vector length is (last index - first index + 1). If the given indices do not match any configured topic of type Analog Value, a SCIL error is returned. See also the data read examples below.

If the format is 8 bits, the order of the bytes within a word is not the same as the transmission order. The first value in SCIL is the lower byte of the register and in the response message it is the second byte of the register. Correspondingly, the second value in SCIL is the higher byte of the register and in the response message, it is the second byte of the register. This applies to all registers in a message. The maximum amount values returned to SCIL is 114 for topic formats using 8 bits or 16 bits and 57 for topic formats using 32 bits. If more registers are needed, the attribute AD (Additional Data) can be used. One modbus response message can contain a maximum of 125 registers. The starting PLC register address of the read request is calculated using the information defined for the topic as follows:

8 bit format -> PLC address = Base Address + (first index - FirstObjectAddress)/2 (rounded downwards)
16 bit format -> PLC address = Base Address + (first index - FirstObjectAddress)
32 bit format -> PLC address = Base Address + (first index - FirstObjectAddress)*2

If AV is written, the type of the topic must be 4 (analog setpoint) and a message of func 6 = Write Single register
or func 16 = Write Multiple registers is formulated. The indices are:

first index = First object address
last index = Last object address

The amount of registers written is dependent on the format of topic:

8 bit format -> (last index - first index + 1)/2 registers is written
16 bit format -> (last index - first index + 1) registers is written
32 bit format -> (last index - first index + 1)*2 registers is written

First and last index of the AV writing need not to be the same as the object addresses defined for the topic. If the given indices do not match any topics of type 'Analog Setpoint', a SCIL error is returned. See also the data write examples below.

If the format is 8 bits, the order of the bytes within a word is not the same as the transmission order.
The first value from SCIL is the lower byte of the register and it is transmitted after the higher byte. Correspondingly, the second value from SCIL is the higher byte of the register and it is transmitted before the lower byte. The maximum amount values given from SCIL are 123 for topic formats using 8 bits or 16 bits and 61 for topic formats using 32 bits. The starting PLC register address of the write request is calculated using the information defined for the topic as follows:

PLC address = Base Address + (first index - FirstObjectAddress)

Access: R/W

Index range: 0..65535

Table continues on next page

AV	Analog Value
Value range:	-2147483647..2147483647 (for topic types other than float) The value range in the message is limited according to the format of the topic

Example 1
Topic is defined to be of type Analog Value with format 4 - one register from PLC's memory. The addresses are FirstObjectAddress = 600, LastObjectAddress = 620 and BaseAddress = 1000.
@x=STA1:SAV(600..609) ; read 10 registers starting from PLC address 1000 to variable x
@x=STA1:SAV(602..609) ; read 8 registers starting from PLC address 1002 to variable x

Example 2
Topic is defined to be of type Analog Setpoint with format 4 - one register from PLC's memory. The addresses are FirstObjectAddress = 500, LastObjectAddress = 520 and BaseAddress = 2000.
#set SAV1:SAV(500..509)=(80,80,80,80,80,80,80,80,80,80) ; write value 80 to 10 registers starting from PLC address 2000
#set SAV1:SAV(502..509)=(80,80,80,80,80,80,80,80) ; write value 80 to 8 registers starting from PLC address 2002

Example 3
Topic is defined to be of type Analog Value with format 8 - floating point. The addresses are FirstObjectAddress = 700, LastObjectAddress = 720 and BaseAddress = 1500. One floating point value requires two registers.
@x=STA1:SAV(700..709) ; read 20 registers (10 floating values) starting from PLC address 1500 to variable x
@x=STA1:SAV(702..709) ; read 16 registers (8 floating values) starting from PLC address 1504 to variable x

Example 4
Topic is defined to be of type Analog Setpoint with format 8 - floating point. The addresses are FirstObjectAddress = 800, LastObjectAddress = 820 and BaseAddress = 2500. One floating point value requires two registers
#SET STA1:SAV(800..809) = (80.1,80.1,80.1,80.1,80.1,80.1,80.1,80.1,80.1,80.1) ; write value 80.1 to 20 registers starting from PLC address 2500
#SET STA1:SAV(802..809) = (80.1,80.1,80.1,80.1,80.1,80.1,80.1,80.1) ; write value 80.1 to 16 registers starting from PLC address 2502

DV	Digital Value

This attribute refers to the register data in the PLC memory.
If the data is requested using only the DV-attribute, the topic interval must be set to 0. Unlike the attribute AV, DV does not support data formats of 8 bits and 32 bits.

If DV is read, the type of the topic must be 9 (digital value) and the returned value is a vector of values read from the registers of the PLC. The meaning of the indices are:

First index = First object address
Last index = Last object address

The first and last index of the DV reading do not need to be the same as the defined object addresses for the topic.
Returned vector length is (last index - first index + 1). If the given indices do not match any configured topic of the type Digital Value, a SCIL error is returned. See also the data read examples below.

The maximum amount value returned to SCIL is 114. If more registers are needed, the attribute AD (Additional Data) can be used. One Modbus response message can contain a maximum of 125 registers. The starting PLC register address of the read request is calculated using the information defined for the topic as follows:

PLC address = Base Address + (first index - FirstObjectAddress)

If AV is written, the type of the topic must be 3 (digital setpoint) and a message of func 6 = Write Single register or func 16 = Write Multiple registers is formulated.
The indices are:

first index = First object address
last index = Last object address

The register count written is (last index - first index + 1).

The first and last index of the DV writing do not need to be the same as the defined object addresses defined for the topic. If the given indices do not match any topics of the type Digital Setpoint, a SCIL error is returned. See also the data write examples below.

The maximum amount value given from SCIL is 123.
The starting PLC register address of the write request is calculated using the information defined for the topic as follows:

PLC address = Base Address + (first index - FirstObjectAddress)

Table continues on next page

DV	Digital Value

Access: R/W

Index range: 0...65535

Value range: 0...65535

Example 1
Topic is defined to be of the type Digital Value with format 5 - one register from PLC's memory (unsigned). The addresses are FirstObjectAddress = 700, LastObjectAddress = 800 and BaseAddress = 600.
@x=STA1:SDV(700..799) ; read 100 registers starting from PLC address 600 to variable x
@x=STA1:SDV(702..799) ; read 98 registers starting from PLC address 602 to variable x

Example 2
Topic is defined to be of the type Digital Setpoint with format 5 - one register from PLC's memory (unsigned). The addresses are FirstObjectAddress = 100, LastObjectAddress = 110 and BaseAddress = 300.
#SET STA1:SDV(100..109)=(80,80,80,80,80,80,80,80,80,80) ; write value 80 to 10 registers starting from PLC address 300
#SET STA1:SDV(102..109)=(80,80,80,80,80,80,80,80) ; write value 80 to 8 registers starting from PLC address 302

GD	General Request of Data

This attribute is not exactly a communication attribute. It is a request to send all the internal PLC NET database data to SYS600. It does not perform any communication between the NET and a remote PLC. When data is sent, the PLC station sends the system's status message to SYS600. When reading, the value of GD is 1 until the update is ready.

Access: R/W

Index range: none

Value range: 0...1

SY	Synchronize

The SY attribute is used to accurately time-synchronize the Modbus device. Currently REF 542plus is supported. The sent time is taken from the internal clock of the MicroSCADA X computer controlling the Modbus line. A topic of type 102 (REF 542plus time sync) must be configured before this function can be used.

Data type: Integer

Value: 1 = time sync for REF 542plus (no other values supported)

Access: Write-only

When events are polled using the topic type 101 from the REF 542plus device, the corresponding static data needs to also be requested in order to ensure that the database is in a consistent state. The maximum value for the static data polling of a input data is currently 65 seconds, which can be too small in some situations and result in useless traffic in the serial line. A new attribute is presented to give an alternate way to initiate the static data transmission.

GI	General Interrogation

This attribute can be used to request all configured input topics in the following way: If the topic is normally polled, meaning that its interval is > 0, the polling is speeded up. If the topic is not polled, meaning that its interval = 0, it is requested if the bit 0 of the RM attribute is 1.

Data type: Integer

Value: 1 = (no other values supported)

Access: Write-only

The response data updates the process database although no changes have occurred. This attribute can also be used when event polling is not used.
It is possible that the interval parameter of the input topics is set to 0 for some topics and the data is requested using the SI, DI, AV and DV attributes. These topics are not included in the GI sequence, if they are not explicitly configured with an RM attribute bit 0.

MM	Transparent Modbus Message

Modbus messages can be sent using this attribute. The responses can be accessed by reading the attribute. The reading can only be done once, which is why it should be stored into a variable in SCIL. If in a response data length is larger than 228 bytes (113 registers + function code and length bytes) the response is stored into separate MM attributes indices. The first 228 bytes are stored into index 1 and the rest into index 2. The indices can be read separately.

Access:	R/W
Index range:	Writing: Not used Reading: 0-1 = Read index 1 2 = Read index 2
Value:	Writing: Byte array without checksum, request max length 200 bytes

Example:
Writing:
#SET STA1:SMM=(10,1,12,22,23)
Reading index 1:
@RESPONSE1 = STA1:SMM1 (or @RESPONSE = STA1:SMM)
Reading index 2:
@RESPONSE2 = STA1:SMM2
The first byte in the vectors is always the modbus function code, but the meaning of the rest of the bytes depends on the function code used. The detailed structure of the each message is specified in the Modbus protocol available from www.modbus.org.
The structure of the most common reading message (Function Codes 01,02,03 and 04) is:
(Function Code, Start Address high byte, Start Address low byte, Quantity high byte, Quantity low byte)
and the corresponding response vector is (Function Code, Byte Count, First data high, First data low, Second data high, Second data low, ...)

AD	Additional Data

This attribute should be read when the response vector to the previous reading of the AV, DV, DI or SI attributes is shorter than expected.
The response to the AD attribute can be a vector of items of the same type as the previous response to the reading of the AV, DV, SI or DI attributes.
The response to the AD reading is formed, if the answer from the device does not fit into one internal message. No new request is made to the device when AD is read.
If a new AV, DV, DI or SI request is made or an AD is read, the stored AD response is cleared.
Error 13832 PLCC_NO_ADDITIONAL_DATA_AVAILABLE is returned to SCIL, if no data is stored.

Indexing:	No
Access:	Read only
Value range	0..0xFFFFFFFF, range depends on the preceding request

Example
Attribute read STA5:SDV(3001..3125) returns a vector of 114 words. Because this is less than expected, the SCIL application should branch to make an additional read of the AD attribute (STA5:SAD, no indexing). This returns a vector of 11 words. These two vectors may be concatenated by the SCIL application if needed.
The application should not make a new read of DV, AV, SI or DI before the AD attribute is read, otherwise the data for AD is lost. This can be a problem in the rare case of multiple SCIL instances accessing the same STA object at the same time.

NP	Next Poll

This attribute can be used to speed up or slow down the polling interval of a topic, generate demand scans, and enable or disable the polling of a single topic. This attribute is useful if there is a need to control the polling of the IED from the SCIL application. The given interval can be configured to be permanent. The given index of the NP attribute must be the same as the one given in the topic definition using the attribute TP.

Data type:	Integer or a vector of two integers
Value:	Vector (INTERVAL, [PERMANENT])
Index:	Topic index

Table continues on next page

NP	Next Poll
Access:	Write only
	INTERVAL Defines the time in milliseconds to the next poll of the topic defined with topic index. Value = 1 generates an immediate poll even if the permanent interval is 0 (=polling disabled). Delayed polling is not possible if the polling is disabled.
	PERMANENT A non-zero (use 1) value defines the given interval to be permanent for the topic. If interval = 0 is set as permanent, the topic is not polled anymore.

Examples:

```
#SET STA'n':SNP'index' = (0,1) ;disable the polling of the topic
#SET STA'n':SNP'index' = 1 ;poll the topic once (demand scan)
#SET STA'n':SNP'index' = (20000,1) ;enable the polling of the topic
using 20 second interval
#SET STA'n':SNP'index' = (3000,0) ;speed up the polling for one poll
#SET STA'n':SNP'index' = (3000,1) ;redefine the polling interval to be
3 seconds
```

5.3.2 Example of using device interface commands

The SYS600 application program can write data to an external device through the process database (SYS600 generates a process message automatically, when the output type of an RTU process object is updated). Another method of sending commands from SYS600 is to use communication attributes as described in the following sections.

5.3.2.1 Writing object commands

Object commands (e.g. switching device open/close commands) are sent as control relay output block messages. This message is a multi-purpose command. This section gives an example of how to write object commands.

Topic configuration

```
#SET STA1:STP(5)=(1,1,30,1,0,1,0,0)
```

Process object configuration			
Name:	PLC_1_OC	Station:	1
Index:	2	OA:	2
Type:	Object command		

The following commands set the same binary output (OUT_BIT1) in the PLC:

```
#SET PLC_1_OC:P2 = 1
#SET STA1:SSI(2) =1 ;-> index = object_address
```

5.3.2.2 Writing analog setpoints

Process object configuration			
Name:	PLC_1_AS	Station:	1
Index:	1	OA:	1
Type:	Analog setpoint		

If 16-bit values are used:
Topic Configuration

```
#SET STA1:STP(7)=(1,1,3,4,313,4,0,0)
```

The following commands set the register value 314 to 1234 in the PLC:

```
#SET PLC_1_AS:P1 = 1234
#SET STA1:SAV(1) = 1234
```

The following command updates a vector in the PLC (reg 314 = 1234, reg 315 = 1, reg 316=2)

```
#SET STA1:SAV(1..3) = (1234,1,2)
```

If 32-bit values are used:
Configuration

```
#SET STA1:STP(7)=(1,1,3,4,313,6,0,0)
```

The following commands sets the register value 314 to 1 and the register value 315 to 57920 in the PLC:

```
#SET PLC_1_AS:P1 = 123456
#SET STA1:SAV(1) =123456
```

If the topic format is 7 (MSB_LONG), the values in the registers are used in opposite order.

If float values are used:
The configuration is:

```
#SET STA1:STP(7)=(1,1,3,4,313,8,0,0)
```

The following command sets the value 3.21 to registers 314, 315

```
#SET STA1:SAV(1)=3.21
```

5.3.2.3 Writing digital setpoints

Process object configuration:

Name:	PLC_1_DS	Station:	1
Index:	1	OA:	1
Type:	Digital setpoint		

Topic configuration

```
#SET STA1:STP(6)=(1,1,3,3,253,5,0,0)
```

The following commands set the register value 254 to 1234 in the PLC:

```
#SET PLC_1_DS:P1 = 1234
#SET STA1:SDV(1) =1234
```

Updating a vector (reg 254 = 1234, reg 255 = 1, reg 256=2)

```
#SET STA1:SDV(1..3) = (1234,1,2)
```

5.3.3 Configuration examples

As previously described, the NET continuously scans the PLC memory. Only data changes are sent to SYS600. The PLC station sends data to the SYS600 process database.

The process database object address is calculated as:

OA = ((PLC_address - Topic_Base_Address) + StartOA)+4096*Topic_Type

It is also possible to read data from the PLC using communication attributes as described in the next sections.

5.3.3.1 Reading indications

Process object configuration:

Name:	PLC_1_SI	Station:	1
Index:	1	OA:	1
Type:	Indication	OB:	0

Topic configuration

```
#SET STA1:STP(1)=( 1,1,1,7,0,1,1000,0)
```

Scanning

The NET reads the state of 16 input bits starting from BIT0 with an interval of 1 second. The process value PLC_1_SI:P1 is updated, if the state of BIT0 of PLC is changed.

The direct reading of the INPUT BIT0 state is possible with the following SCIL command:

```
@BIT = STA1:SSI(16) ;-> index = 16*objcet_address + bit_address
```

and reading vector (16 values):

```
@BIT_V = STA1:SSI(16..31)
```

5.3.3.2 Reading analog values

Process object configuration

Name:	PLC_1_AV	Station:	1
Index:	1	OA:	1
Type:	Analog value		

Topic configuration

```
#SET STA1:STP(3)=(1,1,20,6,199,4,10000,10)
```

Scanning

The NET unit reads 20 registers from the PLC starting from register 200 with an interval of 10 seconds. The process value PLC_1_AV:P1 is updated, if the register 200's value is changed.

The direct reading of the register 200 value is possible with the following SCIL command:

```
@VALUE = STA1:SAV(1)
```

and reading a vector:

```
@VALUE = STA1:SAV(1..20)
```

5.3.3.3 Reading digital values

Process object configuration			
Name:	PLC_1_DV	Station:	1
Index:	1	OA:	1
Type:	Digital value		

Configuration

```
#SET STA1:STP(2)=(1,1,2,9,399,4,1000,000)
```

Scanning

The NET reads two registers from the PLC memory starting from register 400 with an interval of 1 second.

The process value PLC_1_DV:P1 is updated, if the register 400 is changed. The following SCIL command reads the value of register 400 from the PLC:

```
@VALUE = STA1:SDV(1)
```

and reading a vector:

```
@VALUE = STA1:SAV(1..2)
```

5.3.3.4 Reading digital values from input registers

Process object configuration			
Name:	PLC_1_IR	Station:	1
Index:	1	OA:	1
Type:	Digital value		

Configuration

```
#SET STA1:STP(2)=(1,1,2,9,399,10,1000,0)
```

Scanning

The NET reads two input registers from the PLC memory, starting from register 400 with an interval of 1 second.

The process value PLC_1_DV:P1 is updated, if the register 400 is changed. The following SCIL command reads the value of input register 400 from the PLC:

```
@VALUE = STA1:SDV(1)
```

5.3.4 Modbus TCP/IP protocol configuration examples

A setup of one station:

```
@NET=2

@STA = 1
@LINE = 1
#SET NET'NET':SPO'LINE'=25
#SET NET'NET':SDV(28)=('STA','LINE')
#SET NET'NET':SSD'LINE'="TCP"
#SET NET'NET':SEN'LINE'=3
#SET NET'NET':SLD'LINE'="10.0.10.1"
#SET NET'NET':SMS'LINE'='NET'
#SET NET'NET':SPM'LINE'=0
```

```
#SET NET'NET':SIU'LINE'=1
#SET NET'NET':SIU'LINE'=0

#SET STA'STA':SAL=1
#SET STA'STA':SSA=1
#SET STA'STA':SIA="10.0.10.10"
#SET STA'STA':SET=3
#SET STA'STA':SIU=1
#SET NET'NET':SIU'LINE'=1
```

A setup of three stations:

```
@NET=2
@LINE = 1

#SET NET'NET':SPO'LINE'=25
#SET NET'NET':SDV(28)=(1,'LINE')
#SET NET'NET':SDV(28)=(2,'LINE')
#SET NET'NET':SDV(28)=(3,'LINE')
#SET NET'NET':SSD'LINE'="TCP"
#SET NET'NET':SEN'LINE'=3
#SET NET'NET':SMS'LINE'='NET'
#SET NET'NET':SPM'LINE'=0
#SET NET'NET':SIU'LINE'=1
#SET NET'NET':SIU'LINE'=0

@STA = 1

#SET STA'STA':SAL=1
#SET STA'STA':SSA='STA'
#SET STA'STA':SIA="10.0.10.101"
#SET STA'STA':SIU=1

@STA = 2

#SET STA'STA':SAL=1
#SET STA'STA':SSA='STA'
#SET STA'STA':SIA="10.0.10.102"
#SET STA'STA':SIU=1
@STA = 3

#SET STA'STA':SAL=1
#SET STA'STA':SSA='STA'
#SET STA'STA':SIA="10.0.10.103"
#SET STA'STA':SIU=1

#SET NET'NET':SIU'LINE'=1
```

5.3.4.1 TCP additions for LAN/WAN network

If not explicitly specified with line attribute LD, SYS600 Modbus TCP uses the default IP address provided by the operating system.

The created Modbus TCP line reserves a port number for its internal use:

2501+linenumber

In TCP/IP mode (connection-oriented), the connection is established to the standardized port of the slave device:

502

The slave device must accept connections from this port. If not explicitly specified, only one connection to each Modbus TCP server is established at the same time. The second example below describes the gateway configuration. The IP address of the slave is configured with the IA attribute of the station object. In case the Modbus TCP server uses a special TCP port number instead of the standardized one, the device attribute IA provides a special notation for the definition of the port.

5.3.4.2 Example topologies:

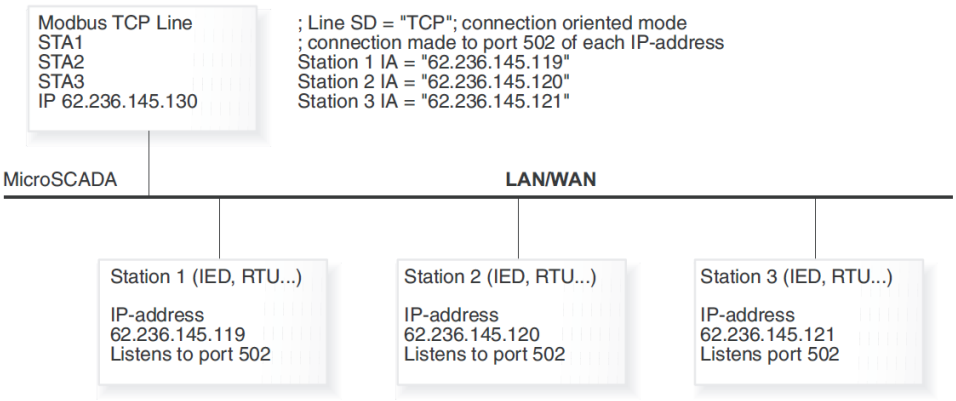


Figure 14: SYS600 as Modbus TCP master in multidrop environment

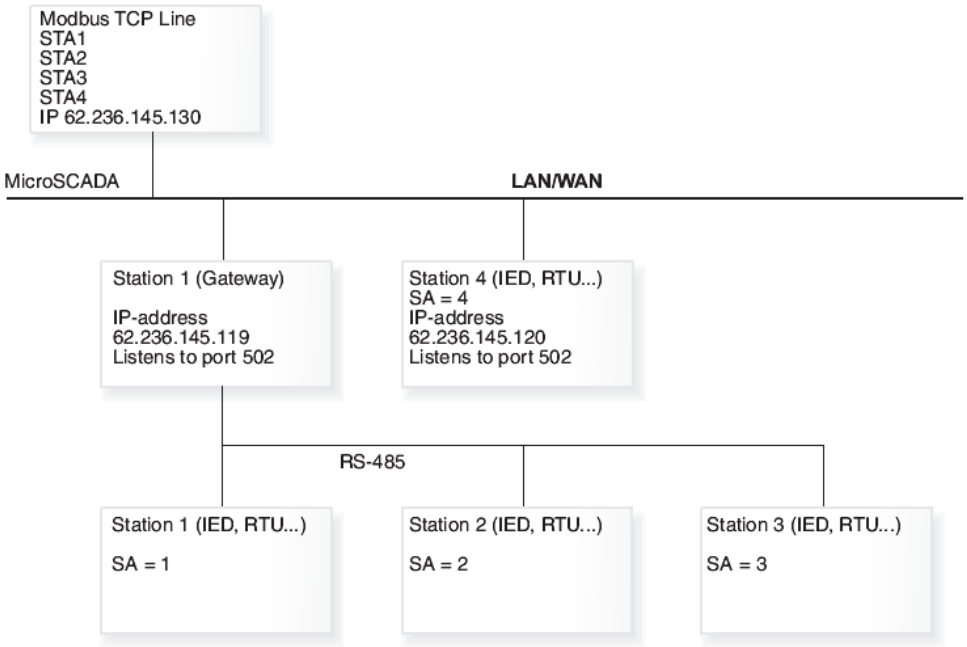


Figure 15: SYS600 as Modbus TCP master in multidrop environment with gateway

Example 1

;connection made to port 502 of each IP-address

;one TCP session established to address 62.236.145.119 (most common case)

Station 1 IA="62.236.145.119"

Station 2 IA="62.236.145.119"

Station 3 IA="62.236.145.119"

Station 4 IA="62.236.145.120"

Example 2

;connection made to port 502 of each IP-address

;three concurrent TCP sessions established to address 62.236.145.119

Station 1 IA="62.236.145.119"

Station 2 IA="62.236.145.119:2"

Station 3 IA="62.236.145.119:3"

Station 4 IA="62.236.145.120"

5.4 Interoperability list

Functions and parameters:

☒ **Supported** by SYS600 Modbus Master

☐ **Not supported** by SYS600 Modbus Master

5.4.1 Network Configurations

☒ Point-to-point

☒ Multipoint star

☒ Multiple point-to-point

5.4.2 Physical Layer

Electrical Interface

☒ RS-232

☒ RS-485

☒ Ethernet

5.4.3 Link Layer

Protocol Mode

☒ ASCII

☒ RTU

Message Length

255 Maximum length L (number of characters)

5.4.4 Transmission Speed (only serial)

- [X] 300 bit/s
- [X] 600 bit/s
- [X] 1200 bit/s
- [X] 2400 bit/s
- [X] 4800 bit/s
- [X] 9600 bit/s
- [X] 19200 bit/s
- [X] 38400 bit/s
- [X] 57600 bit/s

5.4.5 Transmission Settings (only serial)

Parity Check

- [X] No parity check
- [X] Even parity check
- [X] Odd parity check

Stop Bits

- [X] 1 stop bit
- [X] 2 stop bits

5.4.6 Application Layer

Function Codes

- [X] <1> Read Coil Status
- [X] <2> Read Input Status
- [X] <3> Read Holding Register
- [X] <4> Read Input Registers
- [X] <5> Force Single Coil
- [X] <6> Write Single Register
- [ ] <7> Read Exception Status
- [X] <8> Diagnostics
- [ ] <11> Get Communication Event Counter
- [ ] <12> Get Communication Event Log
- [X] <15> Force Multiple Coils
- [X] <16> Write Multiple Registers
- [ ] <17> Report Slave ID
- [ ] <20> Read File Record
- [ ] <21> Write File Record
- [ ] <22> Mask Write Register
- [ ] <23> Read/Write Multiple Registers
- [ ] <24> Read FIFO Queue

PLC Device communication attribute MM Transparent Modbus Message may provide a SCIL solution for unsupported function codes. See attribute description for details.

Appendix A Appendix: Serial cable wiring diagram

When connecting the Modbus master to a SYS600 slave using a direct serial cable, the wiring illustrated by the following figure can be used:

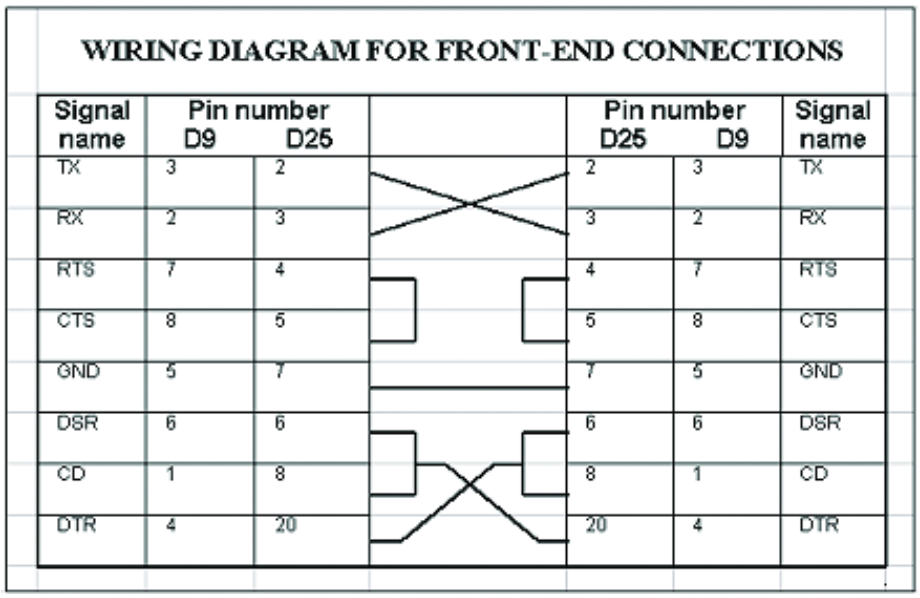


Figure 16: Serial cable wiring diagram

Index

Hitachi ABB Power Grids
Grid Automation Products
PL 688
65101 Vaasa, Finland

https://hitachiabb-powergrids.com/microscadax

Scan this QR code to visit our website

1MRK 511 497-UEN