

# PL7 Micro/Junior/Pro

## Communication applications

### Volume 1

TLX DS COM PL7 xx eng



## Document set

---

### At a Glance

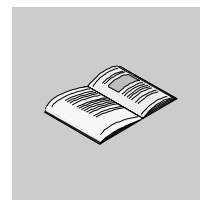
This manual consists of three volumes:

- Volume 1
    - Common communication function
    - Nano transport PLCs
    - Communication in character mode
    - Communication by Uni-telway bus
  - Volume 2
    - Communication by Modbus
    - Communication by Modem
    - Communication by Modbus plus
    - Communication by FIPIO bus
  - Volume 3
    - Communication on FIPWAY network
    - Communication on ETHERNET network
    - Multi-network architecture
-



---

# Table of Contents



<b>About the book</b>	<b>13</b>
<b>Part I Common communication function</b>	<b>15</b>
presentation.	15
<b>Chapter 1 General points about the operations function Communication</b>	<b>17</b>
Presentation	17
Presentation of the operation Communication	18
How to implement a message	19
<b>Chapter 2 Addressing</b>	<b>21</b>
Presentation	21
2.1 General points	22
Presentation	22
Addressing of a communication item	23
Addressing system with PL7 language	24
Type of addressing	25
Addressing a processor's channels of communication	26
Addressing a TSX SCY 21600/21601 communication module	28
Examples of intra-station addressing : Uni-telway addressing	29
Examples of intra-station addressing : FIPIO addressing	31
Inter-station Addressing Examples	32
2.2 Communication via a programming terminal.	35
Presentation	35
Communicating via a programming terminal.	36
How to define the PLC's address	37
Examples of connection in Uni-telway mode.	38
Examples of remote connection in FIPIO or FIPWAY mode.	40
Examples of remote connection in Uni-telway mode	42
Examples of remote connection in ETHWAY mode or in TCP/IP mode	44
<b>Chapter 3 Communication functions</b>	<b>47</b>
Presentation	47
3.1 General points	48

---

	Presentation . . . . .	48
	Presentation of the communication tools . . . . .	49
	Structure of communication functions . . . . .	50
	The communication functions . . . . .	51
	Target address . . . . .	54
	Structure of management parameters . . . . .	55
	Management parameters: communication and operation reports . . . . .	56
	Management parameters : Length and Timeout . . . . .	59
	Performances . . . . .	61
	Server function . . . . .	64
3.2	Help whilst entering the communication functions . . . . .	66
	Presentation . . . . .	66
	Help with entering the communication functions . . . . .	67
	Accessing a specific function, method or procedure type instruction . . . . .	68
	Help on address input . . . . .	70
3.3	Description of the different communication functions . . . . .	72
	Presentation . . . . .	72
	Reading standard objects: READ_VAR . . . . .	74
	Reading standard objects : input Help screen . . . . .	77
	Reading standard objects : example of use . . . . .	78
	Reading standard objects : example of network use . . . . .	79
	Reading standard objects : example of exchange of variables with parameter monitoring . . . . .	81
	Reading standard objects : More detail on reading bits . . . . .	83
	Reading standard objects : reading a timer's current parameters . . . . .	85
	Reading standard objects : reading the current parameters of a monostable . . . . .	86
	Writing standard objects : WRITE_VAR . . . . .	87
	Writing Standard Objects: Entry screen help . . . . .	89
	Writing standard objects: example of use . . . . .	90
	Writing standard objects: example of network use . . . . .	91
	Writing standard objects: example of exchange of variables with parameter monitoring . . . . .	93
	Transmitting UNI-TE Requests: SEND_REQ . . . . .	95
	UNI-TE request transmission input Help screen . . . . .	97
	UNI-TE request transmission example of network use . . . . .	98
	UNI-TE request transmission: List of requests . . . . .	100
	Exchange of text data : DATA_EXCH . . . . .	104
	Exchange of text data : input Help screen . . . . .	106
	Exchange of text data : examples of use . . . . .	107
	Text Type Data Exchange: examples of use with an Altivar . . . . .	109
	Sending a telegram : SEND_TLG . . . . .	111
	Sending a telegram: entry help screen . . . . .	112
	Sending a telegram : example of use . . . . .	113
	Receiving a telegram : RCV_TLG . . . . .	114
	Receiving a telegram : example of use . . . . .	116

---

	Writing a string of characters. PRINT_CHAR. ....	117
	Writing a chain of characters : input help screen .....	119
	Writing of a chain of characters : example of use .....	120
	Reading a character string: INPUT_CHAR .....	122
	Reading a string of characters : input Help screen .....	124
	Reading a character string : example of use .....	125
	Sending/receiving a string of characters : OUT_IN_CHAR. ....	126
	Sending/receiving a string of characters : input Help screen .....	128
	Sending/receiving a string of characters : example of use .....	129
	Stopping an exchange whilst in action. : CANCEL .....	131
	Stopping an exchange whilst in action : example of use. ....	132
	Shifting a byte to the right in a table : ROR1_ARB .....	134
	Shifting a byte to the right in a table : example of use. ....	135
	Swapping bytes in a word table : SWAP. ....	137
	Reading of common Modbus Plus data: READ_GDATA. ....	138
	Writing of Modbus Plus common data : WRITE_GDATA .....	139
	Immediate server : SERVER .....	140
	Immediate server : example of use .....	142
	Asynchronous messaging service : WRITE_Asyn and READ_Asyn .....	143
3.4	Characteristics of communication .....	146
	Presentation .....	146
	Specifications for Communication between TSX Micro/Premium and Series 7 .....	147
	General rules for changing between applications .....	149
	General rules for an exchange to a UNI-TE server. ....	151
	Other examples of compatibility .....	152
3.5	Objects linked to communication. ....	154
	Introduction .....	154
	Default exchanges. ....	155
	Specified exchanges : General points .....	157
	Exchange and report management. ....	159
<b>Chapter 4</b>	<b>Configuration of the operations function Communication. .</b>	<b>163</b>
	Presentation .....	163
	Configuration of the communication function .....	164
	Reminders concerning the configuration editor. ....	165
	How to declare a communication module .....	166
	How to declare a communication channel in a processor or module TSX SCY 21600/21601 .....	167
	Description of configuration screens for communication. ....	168
	Description of communication debugging screens .....	170
<b>Part II</b>	<b>Remoting of Nano PLCs .....</b>	<b>173</b>
	Introduction .....	173
<b>Chapter 5</b>	<b>General .....</b>	<b>175</b>
	Introduction .....	175

	Introduction .....	176
	Compatibility .....	177
	Performances: Network Cycle Time: .....	178
	Performance: positioning an output .....	179
	Operating mode .....	182
<b>Chapter 6</b>	<b>Nano PLC remoting services .....</b>	<b>185</b>
	Introduction .....	185
	Exchanging input/output data .....	186
	Exchanging application data .....	187
	Exchanging data with an analog module .....	189
	Contents of %QW words in writing .....	190
	Contents of %IW words on reading .....	192
	Conversion of analog values of input channels .....	194
	Mixed link .....	196
<b>Chapter 7</b>	<b>Configuring remoting of Nano PLCs .....</b>	<b>197</b>
	Introduction .....	197
	How to access parameters of module TSX STZ 10. ....	198
	Configuration screen for remoting Nano PLCs .....	199
	Modbus parameters associated with the application .....	200
<b>Chapter 8</b>	<b>Programming Nano PLC remoting .....</b>	<b>203</b>
	Introduction .....	203
	Example of communication with Nano PLCs .....	204
	Configuring and programming the example .....	205
<b>Chapter 9</b>	<b>Debugging Nano PLC remoting .....</b>	<b>209</b>
	Introduction .....	209
	Debugging screen for remoting Nano PLCs .....	210
	Nano PLC Offset Debugging Screen .....	211
<b>Chapter 10</b>	<b>Language objects associated to Nano PLC remoting .....</b>	<b>213</b>
	Introduction .....	213
	Language object for implicit exchange .....	214
	Explicit exchange language objects .....	215
	Explicit exchange management and reports .....	218
	Language objects associated with configuration .....	219
<b>Part III</b>	<b>Communication using character mode .....</b>	<b>221</b>
	Introduction .....	221
<b>Chapter 11</b>	<b>General .....</b>	<b>223</b>
	Introduction .....	223
11.1	Introduction to communication via character mode .....	224
	Introduction .....	224
	About character mode .....	225



	Flow control . . . . .	226
11.2	Characteristics . . . . .	228
	Introduction . . . . .	228
	Compatibility . . . . .	229
	Performance . . . . .	230
	Operating mode . . . . .	232
<b>Chapter 12</b>	<b>Configuring character mode communication . . . . .</b>	<b>233</b>
	Introduction . . . . .	233
	How to access PCMCIA card parameters in character mode . . . . .	234
	How to access terminal port parameters . . . . .	235
	How to access TSX SCY 21600 / 21601 module parameters . . . . .	236
	Configuration screen in character mode . . . . .	237
	Functions accessible in character mode . . . . .	238
	Parameters in character mode linked to transmission . . . . .	239
	Parameters in character mode linked to message ends . . . . .	241
	Parameters in character mode linked to flow control . . . . .	243
	Additional parameters . . . . .	244
<b>Chapter 13</b>	<b>Programming communication in character mode . . . . .</b>	<b>247</b>
	Available communication functions . . . . .	247
<b>Chapter 14</b>	<b>Debugging communication using character mode . . . . .</b>	<b>249</b>
	Introduction . . . . .	249
	Debugging screen in character mode . . . . .	250
	Debugging parameters in character mode . . . . .	251
	How to test a communication channel . . . . .	253
<b>Chapter 15</b>	<b>Language objects associated with character mode communication . . . . .</b>	<b>255</b>
	Introduction . . . . .	255
	Language objects in implicit exchange . . . . .	256
	Explicit exchange language objects . . . . .	257
	Explicit exchange management and reports . . . . .	259
	Language objects associated with configuration . . . . .	260
<b>Part IV</b>	<b>Communication via Uni-telway bus . . . . .</b>	<b>263</b>
	Introduction . . . . .	263
<b>Chapter 16</b>	<b>General . . . . .</b>	<b>265</b>
	Introduction . . . . .	265
	Introduction . . . . .	266
	Compatibility . . . . .	267
	Performance . . . . .	269
	Operating mode . . . . .	271
	Addresses of a slave PLC . . . . .	272

<b>Chapter 17</b>	<b>Configuring a Uni-telway communication . . . . .</b>	<b>273</b>
	Introduction . . . . .	273
	How to access Uni-telway PCMCIA card parameters . . . . .	274
	How to access terminal port parameters . . . . .	275
	How to access TSX SCY 21600/21601 module parameters . . . . .	276
	Uni-telway link configuration screen . . . . .	277
	Functions accessible using Uni-telway . . . . .	278
	Uni-telway parameters linked to the application . . . . .	279
	Uni-telway parameters linked to transmission . . . . .	281
<b>Chapter 18</b>	<b>Programming a Uni-telway communication . . . . .</b>	<b>283</b>
	Introduction . . . . .	283
18.1	Communication function . . . . .	284
	Introduction . . . . .	284
	Available communication functions . . . . .	285
	Writing command words . . . . .	286
18.2	Master to slave exchange . . . . .	287
	Master to Slave exchanges . . . . .	287
18.3	Slave to master exchange . . . . .	289
	Introduction . . . . .	289
	Slave to Master exchanges . . . . .	290
	Example of a slave exchange with the master system . . . . .	292
	Example of a direct slave exchange with the master system . . . . .	294
18.4	Slave to slave exchange . . . . .	295
	Introduction . . . . .	295
	Slave to Slave exchanges . . . . .	296
	Example of a slave exchange with the slave server . . . . .	298
	Example of a slave exchange with the slave application . . . . .	300
	Example 2 of a slave exchange with the slave system . . . . .	301
	Example of a right shift of 1 byte in a byte table . . . . .	303
	Example of a direct slave exchange with a slave system . . . . .	305
	Example of a slave stopping another slave . . . . .	306
18.5	Event data . . . . .	307
	Event data managed by the master . . . . .	307
<b>Chapter 19</b>	<b>Debugging a Uni-telway communication. . . . .</b>	<b>309</b>
	Introduction . . . . .	309
	Uni-telway debugging screen. . . . .	310
	Uni-telway debugging screen. . . . .	311
	Requests available for the communication channel test . . . . .	312
	How to test a channel with Identification and Mirror requests . . . . .	313
	How to test a channel with requests . . . . .	315
<b>Chapter 20</b>	<b>Language objects associated with Uni-telway communication . . . . .</b>	<b>317</b>
	Introduction . . . . .	317

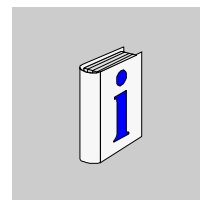
---

20.1	Language objects associated with master Uni-telway mode. . . . .	318
	Introduction . . . . .	318
	Language object for implicit exchange in master Uni-telway mode . . . . .	319
	Language object for explicit exchange in master Uni-telway mode . . . . .	320
	Exchange management and report. . . . .	323
	Language objects associated with configuration in master Uni-telway mode . .	324
20.2	Language objects associated with slave Uni-telway mode . . . . .	325
	Introduction . . . . .	325
	Language objects for implicit exchange in Uni-telway slave mode . . . . .	326
	Language object for explicit exchange in Uni-telway slave mode . . . . .	327
	Exchange management and report. . . . .	329
	Language objects associated with configuration in slave Uni-telway mode . . .	330
<b>Index</b>	. . . . .	<b>331</b>



---

## About the book



---

### At a Glance

**Document Scope** This manual deals with the software implementation of the communication application.

**Validity Note** This updated publication includes the PL7 V4.2 functions.

### Revision History

Rev. No.	Changes
1	Common Remote location of Nano PLCs ASCII mode Uni-telwau bus
2	This version takes in order some OPR
3	This version takes in order some OPRs and a few corrections.

---

### Related Documents

Title of Documentation	Reference Number
Hardware implementation manual	TSX DM 57 42E

**User Comments** We welcome your comments about this document. You can reach us by e-mail at [TECHCOMM@modicon.com](mailto:TECHCOMM@modicon.com)

---



---

# Common communication function



---

## presentation

### Object of this spacer

This spacer presents a general view of the operations function Communication and describes its use with the software PL7.

### What's in this part?

This part contains the following chapters:

Chapter	Chaptername	Page
1	General points about the operations function Communication	17
2	Addressing	21
3	Communication functions	47
4	Configuration of the operations function Communication	163





---

# General points about the operations function Communication

1

---

## Presentation

### Subject of this chapter

This chapter presents a summary of the operations function Communication and its services.

### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Presentation of the operation Communication	18
How to implement a message	19

## Presentation of the operation Communication

---

### Presentation

The communication function allows the exchange of data between all devices connected by a bus or network.

This function applies:

- to the specific communication modules shown on rack,
  - to the processors through the terminal point or cards PCMCIA.
- 

### Types of communication

The different communication functions available are:

- the terminal port function
  - the extension link function to TSX Nano for PLC TSX Micro
  - the character mode function
  - the Uni-telway function
  - the Modbus/Jbus function
  - the Modem function
  - the Modbus Plus function
  - the FIPIO function(manager and agent)
  - the FIPWAY function
  - the ETHERNET (TCP/IP) function
  - the Bridge function
-

---

## How to implement a message

---

### Presentation

The method below describes the principal stages to follow in order to implement a communication function

---

### Method

The procedure is as follows:

Step	Action
1	Determine all the necessary hardware and software components : terminal equipment, branch cables, connection fittings, main cables, wiring testing systems, development systems for process control applications, installation operation systems (adjustment, diagnostics, maintenance).
2	Assemble and test the wiring system (cables and connection fittings).
3	Connect the terminal fittings and configure their communication channels.
4	Test the accessibility of each channel at all architecture points by using PL7 in local mode and in connected mode on both the TSX 57 or TSX 37 present (use of the "transparency" function). This is applicable to PL7-2 or PL7-3 if the TSX 17 or the TSX model 40 is present in the configuration).
5	Program the PLC applications in the same way as their communication functions.

---



---

# Addressing



---

## Presentation

### Subject of this chapter

This chapter presents X-WAY communications principles of addressing.

### What's in this chapter?

This chapter contains the following Sections:

Section	Topic	Page
2.1	General points	22
2.2	Communication via a programming terminal	35

## 2.1 General points

---

### Presentation

**Subject of this chapter**

This sub- chapter presents the communication X-WAY's general rules for addressing.

---

**What's in this section?**

This section contains the following topics:

Topic	Page
Addressing of a communication item	23
Addressing system with PL7 language	24
Type of addressing	25
Addressing a processor's channels of communication	26
Addressing a TSX SCY 21600/21601 communication module	28
Examples of intra-station addressing : Uni-telway addressing	29
Examples of intra-station addressing : FIPIO addressing	31
Inter-station Addressing Examples	32

---

## Addressing of a communication item

**Presentation** In a X-WAY architecture each station is identified by an unique address made up of a network number and a station number.

Inside a station each item of communication also has a topological address which determines its access path.

**Addressing** Addressing is of the form:

$ADR\#\{n.s\}xy.i.j$

The following table shows the different addressing parameters:

Parameter	Description
{n.s}	corresponds to the X-WAY addressing <ul style="list-style-type: none"><li>● n: indicates the number of the network.</li><li>● s: indicates the number of the station.</li></ul>
xy.i.j	corresponds to the topological address <ul style="list-style-type: none"><li>● xy: indicates the number of the rack and of the module respectively (corresponds to the geographical position of the module ).</li><li>● i: indicates the number of the channel.</li><li>● j: indicates the destination device address (slave number, ...).</li></ul>

**Note:** If the number of the rack is other than 0, the number of the module is entered with two digits (examples: 05, 10, ...).

**Example** The example is concerned with slave 2, in channel 1, in the module placed in slot 5 of rack 7 in station 3 on network 20:

$ADR\#\{20.3\}705.1.2$

## Addressing system with PL7 language

---

### Presentation

Some key words are used in order to facilitate access to certain items of communication (server UNI-TE, application PL7,...) or in order to carry out broadcast exchanges.

These exchanges are carried out by communication functions, which are described in the following chapter.

---

### Key words

The key words are:

- the key word `SYS` allows access to UNI-T, which is the server of the CPU, of a channel, of a communication module, ... .
- the key word `APP` allows access to the PL7 application of a station.
- the key word `ALL` is defined to describe a particular type of broadcast. It can replace one of the elements of a topological address. The level of dissemination is determined according to location of the key word `ALL` in the address:
  - joined onto the network number, broadcast occurs in all of the stations in the selected network (example: 2.ALL allows access to all stations connected to network 2),
  - joined onto the station number, broadcasting occurs in all the items linked to the intra-station communication channels (example: 2.4.ALL allows access to all of the communication items in station 4 found on network 2).

**Note:** When the transmitting application wishes to communicate with a text block function in the PL7-2 or PL7-3 application of a PLC TSX series 7, the key word must be `APP.num` where `num` corresponds to the number of the exchanges target text block

---



## Type of addressing

**Presentation** The following table repeats the different types of addressing.

**Local addresses** The local addresses contain the topological addresses and the slave addresses on a bus.

Destination	Address
UNI-TE server of a TSX Micro/Premium	SYS
PL7 application Micro/Junior/Pro	APP
PL7-3 application	APP.number of the text block
Uni-telway slave	module.channel.slave number
Modbus slave	module.channel.slave number
Character mode link	module.channel.SYS
Module server	module.SYS
Sub-module server	module.channel.SYS
Server to a FIPIO device	\\module.channel.connecting point\\SYS

**Distant addresses** Distant addresses correspond to the addresses of equipment connected to a network.

Destination	Address
Target on a distant network	{network.station}local address
Destination on a local network	{station}local address

**Addresses in circulation** The addresses in circulation are functions of the destination equipment.

Destination	Address
Broadcasting to all stations	{network.ALL}local address
Broadcasting to all modules	ALL.SYS
Broadcasting to all Uni-telway or Modbus slaves	module.channel.ALL

## Addressing a processor's channels of communication

### Presentation

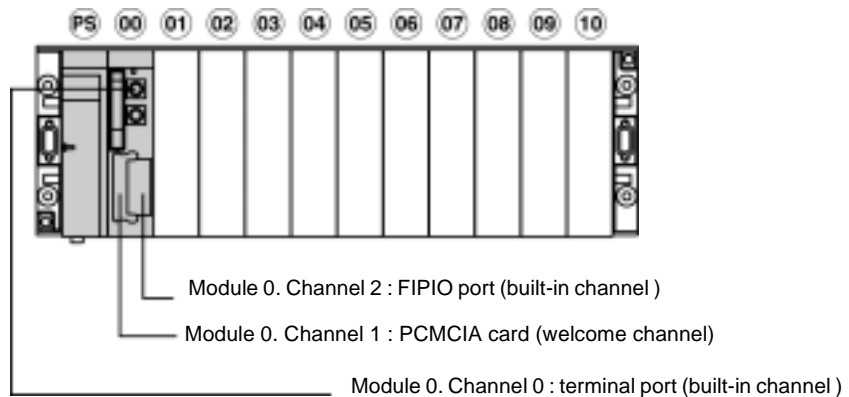
The following examples give the different types of addressing associated with a processor's channels of communication.

The examples will be based upon a TSX Premium processor.

Following the desired configuration, there may be a single or double format supply in the rack which takes up 1 or 2 slots. The modules have a geographic address depending on the position of the module in the rack.

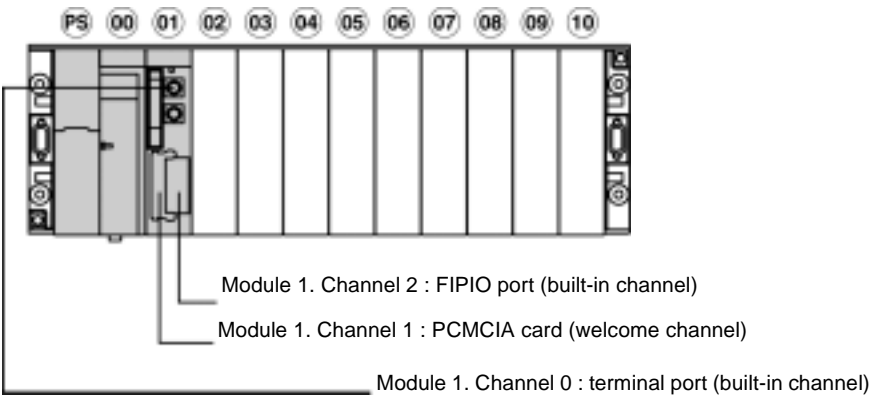
### With a single format supply

The supply takes up a slot. The processor's communication channels will therefore be able to have the following addresses:



**With a single  
format supply**

The supply takes up two slots. The processor's communication channels could therefore have the following addresses:



## Addressing a TSX SCY 21600/21601 communication module

### Presentation

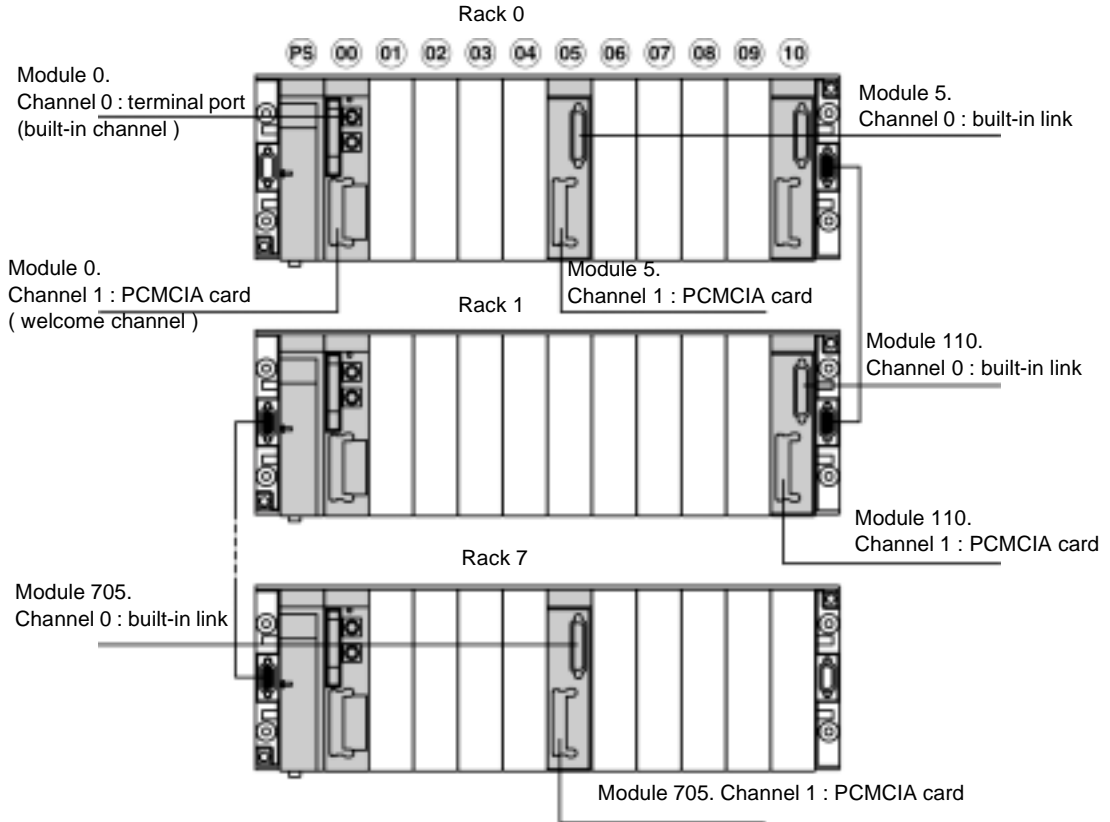
The following examples give the different types of addressing associated with the TSX SCY 21600 and the TSX SCY 21601 communication modules.

The examples will be based upon a TSX Premium processor.

**Note:** Remember that these kind of communication modules are limited in accordance with the type of processor being used. Refer back to the starting manual in order to evaluate the number of communication operations channels.

### Examples

The modules communication channels will be able to have the following addresses:



## Examples of intra-station addressing : Uni-telway addressing

### Presentation

With this addressing, a master station may gain access to different slaves connected on a bus.

In the following examples, the slaves are linked to the master station (station with a TSX Premium processor) by an Uni-telway bus. See Communication via Uni-telway bus, p. 263 .

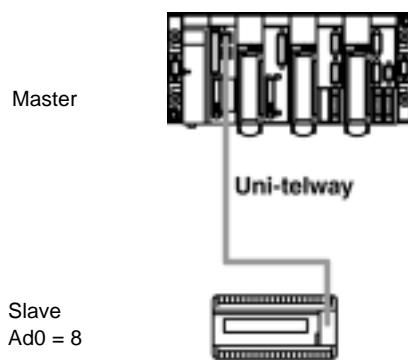
### Addressing rules

Addressing applies in this configuration:

- for the modules address:
  - always 0 if the master station is a TSX Micro PLC
  - 0 to 710 if the master station is a TSX Premium PLC
- for the channel address:
  - 0 if it is a terminal port connection
  - 0 if it is the built in link connection of a TSX SCY 2160 module •
  - 1 if it is a PCMCIA card connection
- for the slave:
  - 1 to 98 if the slave is connected to a PCMCIA card or to the built in link of a TSX SCY 2160 module • .In this case the station master can scan up to 98 slaves.
  - 1 to 8 if the slave is connected to the terminal point. In this case the station master can scan up to 8 slaves.

### Terminal port connection

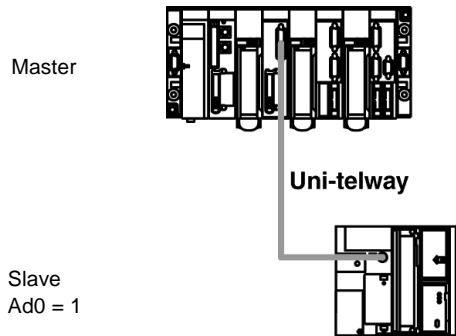
A piece of equipment with the address Ad0=8 is connected to the terminal port of a TSX Premium.



Addressing of slave 8: ADR#0.0.8

**Module TSX SCY  
21600/21601  
connection**

A piece of equipment with the address Ad0=1 is connected to the built in link of a TSX SCY 21600/21601 in position 2 in the rack.



Addressing of slave 1: ADR#2.0.8

---

## Examples of intra-station addressing : FIPIO addressing

### Presentation

Exchanges with the manager are changes of variables or changes of messages.  
See .

To gain access to the server of messaging UNI-TE the syntax of the addressing is as follows:

\module . channel . Linkage point \ SYS

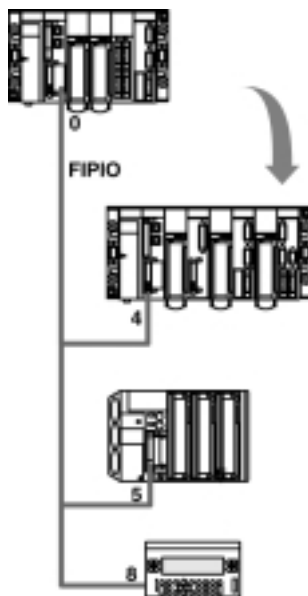
### Addressing rules

In the case of a FIPIO communication, addressing applies:

- for the modules address:
  - 0 if the station master possesses a single format supply
  - 1 if the station master possesses a double format supply
- for the channel address: always 2 because the link is built into the processor
- for the connecting point: 1 to 127 because it is possible to connect up to 127 items of equipment on the bus

### Examples

In the following example, the manager is aimed at TSX Premium at connecting point 4 or CCX 17 at connection point 8.



Addressing of equipment item number 4: \0.2.4\SYS

Addressing of equipment item number 8: \0.20.8\SYS

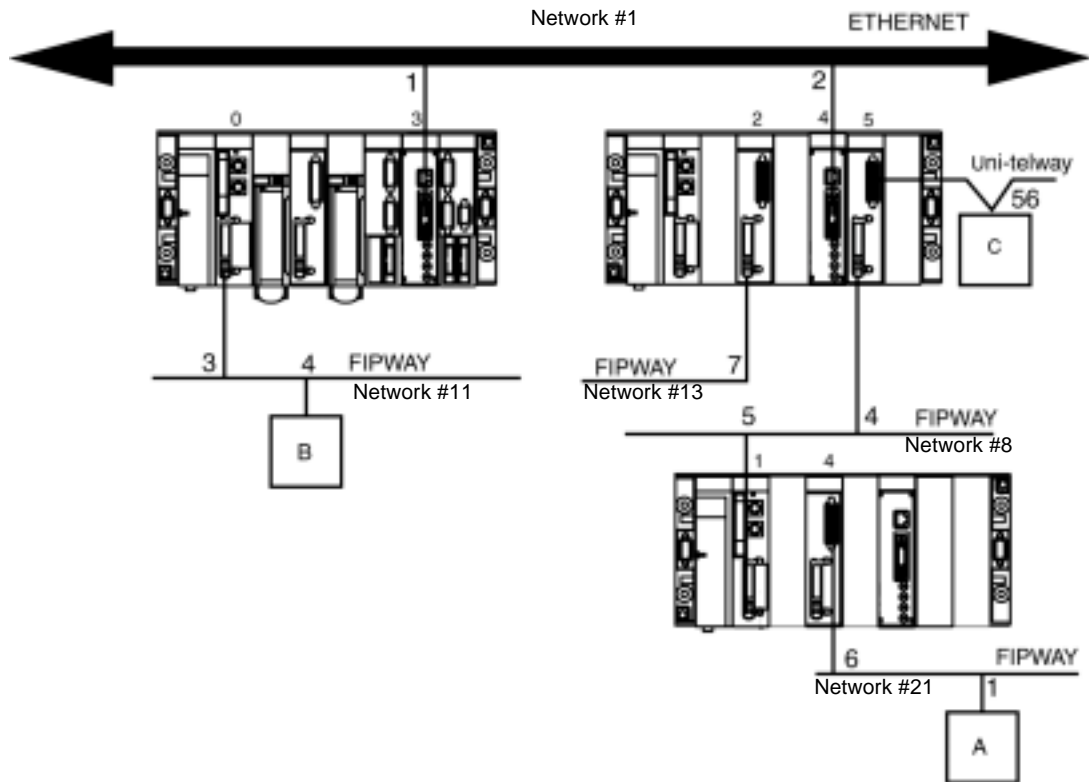
## Inter-station Addressing Examples

### At a Glance

For an inter-station exchange to take place (i.e. an exchange between two stations on the same network or on two different networks), the address must show both the number of the network and the number of the destination entity.

### Example 1

The multi-network configuration is as follows:



In the first case, station B addresses station A's system:

ADR#{21.1}SYS

In the second case, station B addresses station C:

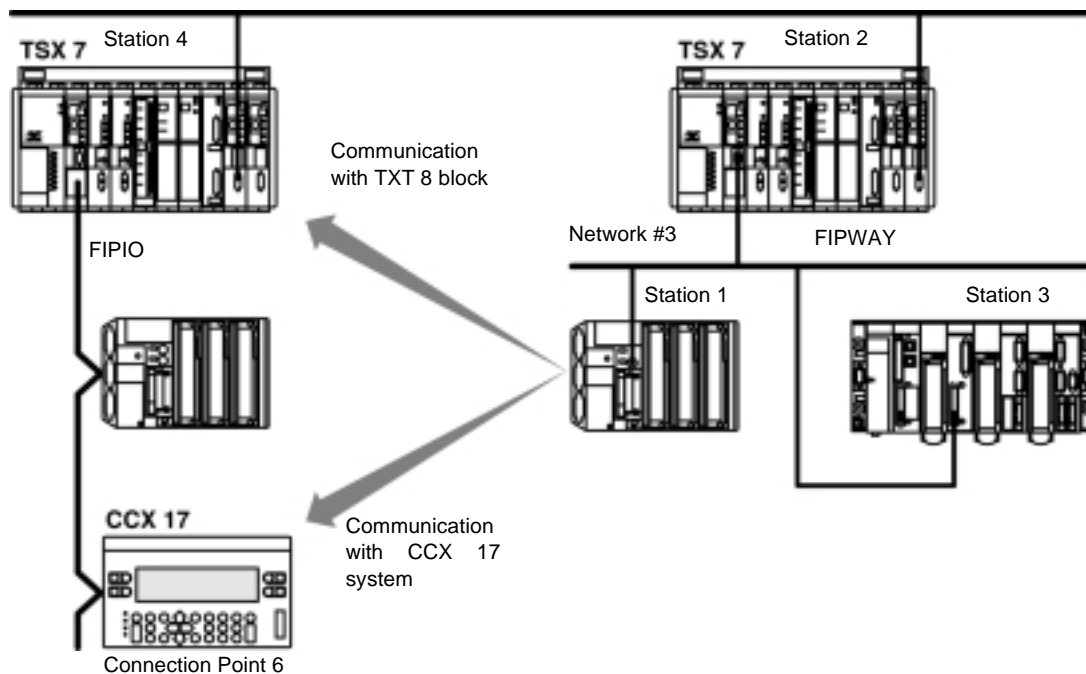
ADR#{1.2}5.0.56



**Example 2**

The example below shows how to access a CCX17 system connected to a FIPIO bus (connection point 6) and communicate with text block TXT 8 on a model 40 programmable PLC on network 2.

FIPWAY: Network #2



The address of the TXT 8 text block on the TSX 7 station 4 PLC is:

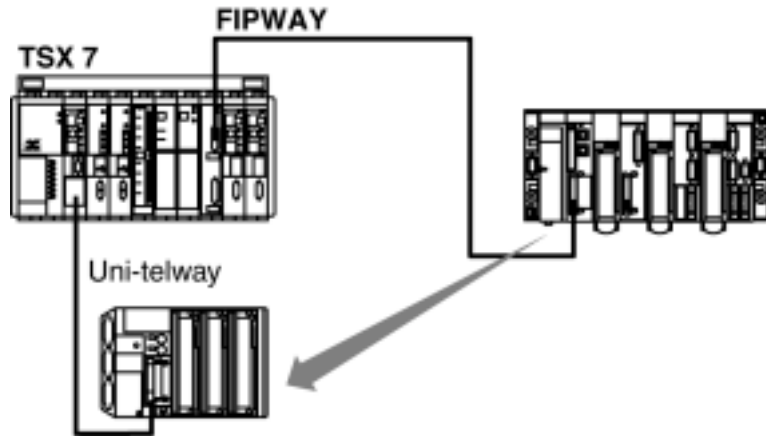
ADR#{2.4}APP.8

The address of the CCX 17 system is:

ADR#{2.4}\0.2.6\SYS

**Example 3**

For a TSX47-107 PLC, the Uni-Telway (SCI) communication module integrated into the processor is accessed by channel 100 = 0.100.x.



The address is:

ADR#{2.4}\0.100.x

---

---

## 2.2 Communication via a programming terminal

---

### Presentation

---

**Subject of this chapter**

This sub-chapter presents the general rules of addressing and communication via a programming terminal.

---

**What's in this section?**

This section contains the following topics:

Topic	Page
Communicating via a programming terminal	36
How to define the PLC's address	37
Examples of connection in Uni-telway mode	38
Examples of remote connection in FIPIO or FIPWAY mode	40
Examples of remote connection in Uni-telway mode	42
Examples of remote connection in ETHWAY mode or in TCP/IP mode	44

---

## Communicating via a programming terminal

---

### Presentation

As for the PLCs TSX Micro and TSX Premium communicating via networks, it is possible to access these stations via the programming terminals.

Access to a remote item of equipment by the programming terminal requires the definition of a driver and of the address of the item of equipment in the PL7 software. This address must be known in order to be able to locate a network PLC.

It is therefore necessary to configure the addresses of each station or item of equipment present on the bus or network before starting to communicate.

The addresses can be configured:

- by the PL7 software for the TSX Micro and TSX Premium PLC,
- in a physical way (example: address coding on the packages TSX SCA 62),
- by specific software.

### Connection possibilities

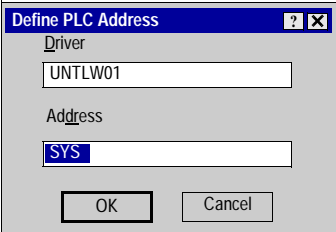
A programming terminal can connect:

- in Uni-telway mode,
- in FIPIO or FIPWAY mode,
- in TCP/IP mode.

## How to define the PLC's address

**Presentation** This procedure allows connection to a remote PLC on a network or a bus.

**Steps to follow** The procedure is as follows:

Step	Action
1	Open PL7 software
2	Select the command <b>AP</b> → <b>Define the address of the PLC...</b>  <b>Result</b> 
3	Select the driver according to the mode of communication.
4	Enter the address of the PLC to be accessed.
5	Validate with the key <b>Ok</b> .

**Observations** The driver `UNTLW01` is chosen by default. It allows access to the stations which are uniquely in Uni-telway mode.  
For a connection in FIPIO mode, select driver `FIO01` or `FIP02`.  
For a connection in ETHWAY mode, select driver `XIP01`

The address by default is `SYS`. This address corresponds to a connection in local mode in order to access the PLC's system.  
For a remote connection, the address must be modified by the remote destination address.

## Examples of connection in Uni-telway mode

---

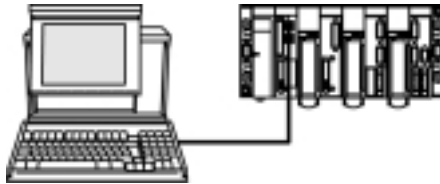
### Presentation

In uni-telway mode, the addresses 1, 2 and 3 are reserved for the programming terminal.

---

### Connection in local mode

The programming terminal is connected on the PLC's TER or AUX port.

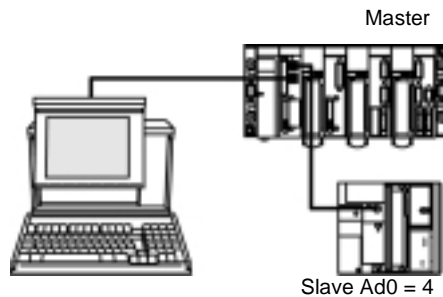


In this configuration, the values by default are appropriate.

---

### Connection to a slave on the terminal port

The PLC slave is connected to the master by the TER port and the programming terminal is linked to the AUX port.

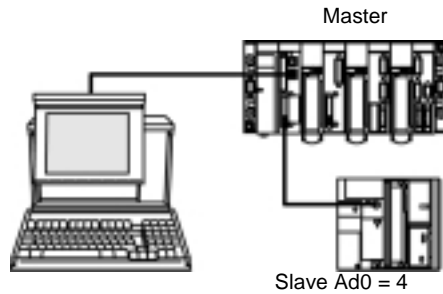


In this configuration, the definition of the PLC's address is as follows:

- the driver is at its value by default.
  - the address is 0, 0, 4.
-

**Connection to a slave on a PCMCIA card**

The PLC slave is connected to the master by the PCMCIA card and the programming terminal is linked to the TER or AUX port.



In this configuration, the definition of the PLC address is as follows:

- the driver is at its value by default.
  - the address is 0, 1, 4.
-

## Examples of remote connection in FIPIO or FIPWAY mode

### Presentation

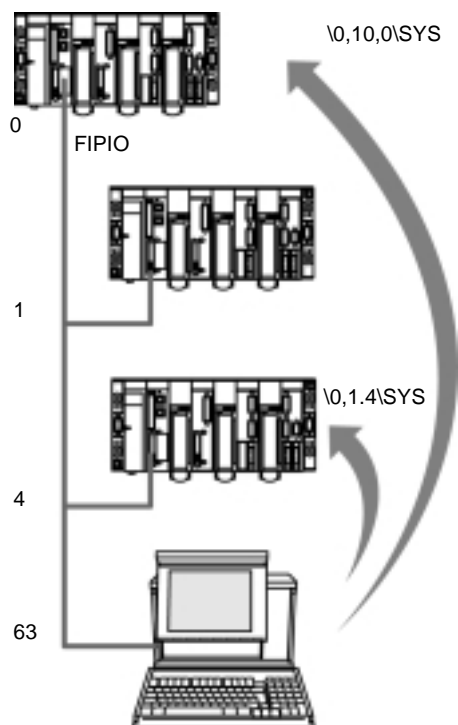
A programming terminal can connect to the PLC in FIPIO mode or in FIPWAY mode.

In both cases, the programming terminal must be fitted:

- either from coupler TSX FPC 10 with the drivers that are linked to it ( each machine equipped with a PC AT bus),
- or from coupler TSX FPP K200 with the drivers that are linked to it ( each machine equipped with a PCMCIA slot),

### Connection in FIPIO mode

In order to communicate in FIPIO mode, the connection port for the programming terminal is 63.



To access station 0, the definition of the PLC address is as follows:

- the driver is at value FIP01 .
- the address is \0,1,0\SYS.

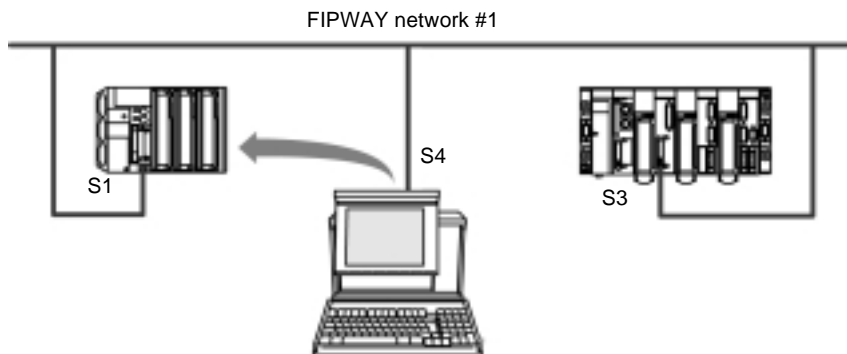


To access station 4, the definition of the PLC address is as follows:

- the driver is at value FIP01 .
- the address is \0,1,4\SYS.

### Connection in FIPIO mode

To communicate in FIPWAY mode, the programming terminal has its own address.



To access station 1, the definition of the PLC's address is as follows:

- the driver is at value FIP01 .
- the address is {1,1}SYS.

## Examples of remote connection in Uni-telway mode

---

### Presentation

The programming terminal accesses the remote stations connected to a network.

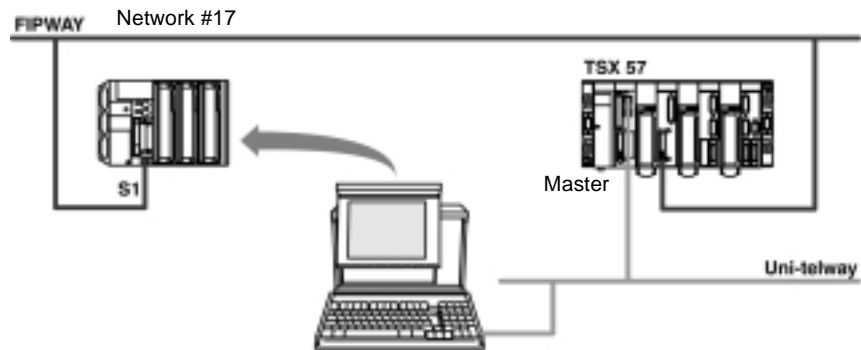
In this configuration, the programming terminal must be connected to a Uni-telway bus. The station master of the Uni-telway bus must be directly linked to the remote station via a network, or linked by interposed networks.

---

### Connection via single network

The main station of the Uni-telway bus is linked directly to the remote station S1 via a FIPWAY network.

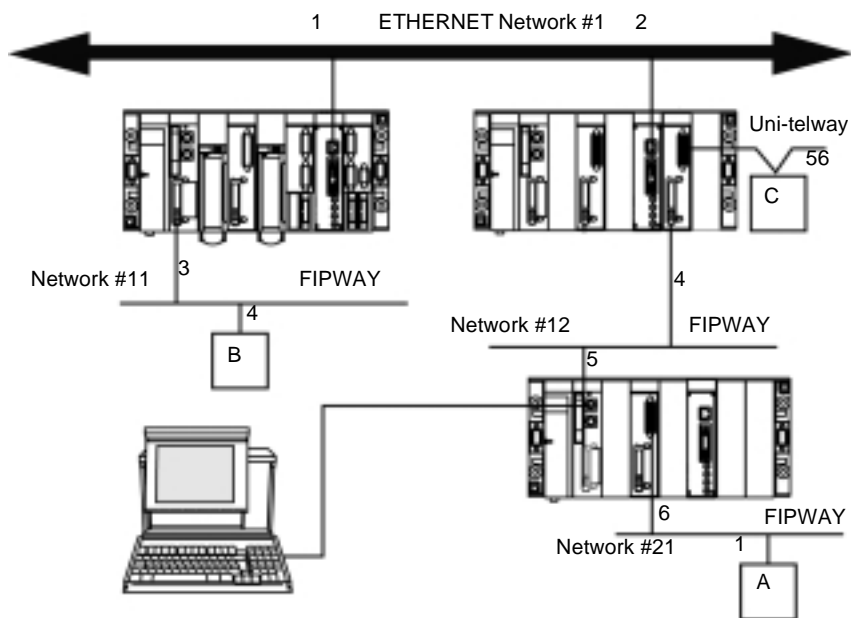
The programming terminal selects the system of the remote station S1.



In this configuration, the definition of the PLC's address is as follows:

- the driver is at its value by default.
  - the address is  $\{17, 1\}_{SYS}$ .
-

The programming terminal is linked to a station in Uni-telway mode and selects systems of stations A and B.



- the driver is at its value by default.
- the address is  $\{21, 1\}_{\text{SYS}}$ .

- the driver is at its value by default.
- the address is  $\{11, 4\}_{\text{SYS}}$ .

## Examples of remote connection in ETHWAY mode or in TCP/IP mode

---

### Presentation

A programming terminal can connect to the PLC in ETHWAY mode or in TCP/IP mode.

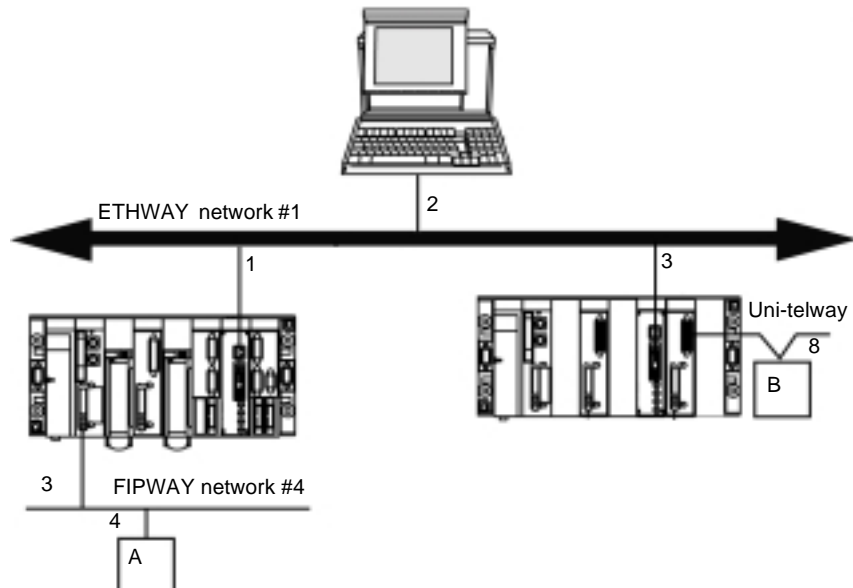
In both cases, the programming terminal is linked to the network via any ETHERNET network card.

**Note:** When the programming terminal is a portable computer, fitted with the operating system for Windows 95 or OS/2, the PCMCIA ETHERNET card can not be used to communicate in ETHWAY mode.

### Connection in ETHWAY mode

To communicate in ETHWAY mode with a programming terminal, the ETHERNET network card is bound to the driver ETHWAY01 or ETHWAY02 (possibility to have two occurrences).

While the driver configures ETHWAY01 it is necessary to provide it with an address {network.station}for the local station.



To access station A, the definition of the PLC's address is as follows:

- the driver is at value `ETHWAY01`.
- the address is `{ 4, 4 }SYS`

To enter station B, the definition of the PLC's address is as follows:

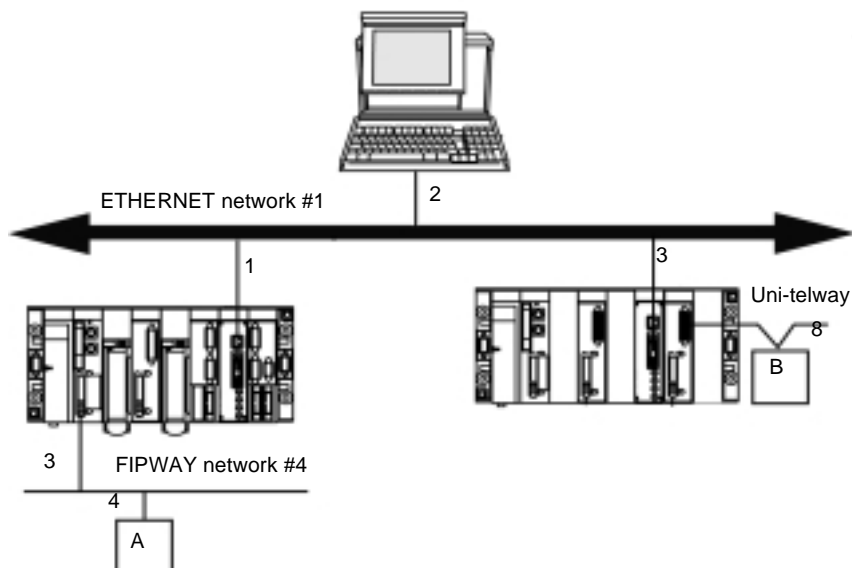
- the driver is at value `ETHWAY01`.
- the address is `{ 1, 3 }5, 0, 8`.

### Connection in TCP/IP mode.

To communicate in TCP/IP mode with a programming terminal, the ETHERNET network card is bound to the driver `XIP01` or `XIP02` (possibility to have two occurrences).

First the XIP driver has the station addresses linked to the programming terminal in its configuration and it must be started before all X-WAY communication on TCP/IP.

While the XIP driver configures itself, it is necessary to provide it with an address {network.station} for the local station.



To access station A, the definition of the PLC's address is as follows:

- the driver is at value `XIP01`.
- the address is `{ 4, 4 }SYS`.

To enter station B, the definition of the PLC's address is as follows:

- the driver is at value `XIP01`.
- the address is `{ 1, 3 }5, 0, 8`.



---

# Communication functions

3

---

## Presentation

### Subject of this chapter

This chapter presents the communication functions

### What's in this chapter?

This chapter contains the following Sections:

Section	Topic	Page
3.1	General points	48
3.2	Help whilst entering the communication functions	66
3.3	Description of the different communication functions	72
3.4	Characteristics of communication	146
3.5	Objects linked to communication	154

## 3.1 General points

---

### Presentation

#### Subject of this sub- chapter

This sub-chapter presents the principle and the structure of the communication functions.

---

#### What's in this section?

This section contains the following topics:

Topic	Page
Presentation of the communication tools	49
Structure of communication functions	50
The communication functions	51
Target address	54
Structure of management parameters	55
Management parameters: communication and operation reports	56
Management parameters : Length and Timeout	59
Performances	61
Server function	64

---



## Presentation of the communication tools

---

### Presentation

All the channels of X-WAY communication use a messaging service to allow the exchange of data.

The PL7 language interface offers specific communication functions which authorize the sending and/or the receipt of messages to / from a communication item.

The destination items of an exchange can be localized just as well in a local station as in a remote station on a communication channel or directly in the CPU.

The communications functions offer an independent interface of the location of the target entity and hide the code of the communication requests from the user. They also guarantee communication compatibility with the TSX model 40 PLC, the TSX 17 and the series 100 PLC.

**Note:** The processing of communication functions is asynchronous with the processing of the applied task which has allowed them to be activated. The only functions that are the exception, are the functions for the sending/receipt of telegrams and the operation shutdown function, because their running is synchronous with the running of the activation task.  
A communication function is said to be synchronous when it is fully implemented during the PLC task that activated it.  
A communication function is said to be asynchronous when it is implemented during one or more PLC tasks after the one that activated it.

## Structure of communication functions

---

### Presentation

A communication function uses:

- an address parameter,
  - parameters specific to a communication operation,
  - management parameters
- 

### Syntax

The syntax of a communication function presents itself in the following form:

Function (Target address, Specific parameters, Management parameters)

The following table describes the different constituent parts of a function:

Part	Description
Function	corresponds to the type of communication function.
Target address	indicates the address of the destination of the exchange.
Specific parameters	depend on the type of communication function. They are described for every communication function.
Management parameters	The management parameters are common to all asynchronous communication functions. They are made up of: <ul style="list-style-type: none"><li>● a parameter that gives information about the activity of the function,</li><li>● a parameter that specifies the exchange number that identifies the transaction in action</li><li>● a parameter that contains the report of the exchange ( communication report and operation report),</li><li>● a timeout parameter that allows the control of the absence of response</li><li>● a length parameter that allows the storage of the number of bytes to send or the number of bytes received.</li></ul>

---

## The communication functions

### Presentation

Functions allow communication from one item of equipment to another. Certain functions are common to several types of communication channel, others can be specific to one communication function.

**Note:** The processing of communication functions is asynchronous in relation to the processing of the application task that has activated them. Only the telegram sending/receipt function and the operation shutdown function are the exceptions because their execution is synchronous with the running of the activation task

### Asynchronous communication functions

A communication function is said to be asynchronous when it is implemented during one or more PLC tasks after the one that activated it.

The following table presents the communication functions with asynchronous application:

Function	Purpose...
READ_VAR	the basic language for reading objects: words, bits, double words, floating internals, constant words, word and bit system, timers, monostables, drum, registers and counters.
WRITE_VAR	the reading of base language objects: word, bit, double word, floating internals, word and bit system.
SEND_REQ	emission of UNI-TE requests
DATA_EXCH	sending and/or request of receipt of data
PRINT_CHAR	writing of a chain of characters.
INPUT_CHAR	reading of a chain of characters.
OUT_IN_CHAR	sending of a chain of characters and wait for a response.
READ_GDATA	reading of Modbus Plus common data
WRITE_GDATA	writing of Modbus Plus common data
SERVER	to handle requests READ_VAR and WRITE_VAR on Modbus immediately (immediate server)..
READ_Asyn	reading of 1 K of electronic messaging.
WRITE_Asyn	the writing of 1 K of electronic messaging.

**Note:** Triggering the asynchronous functions on edge and not on state is recommended.

### Synchronous communication functions

A communication function is said to be synchronous when it is fully implemented during the PLC task that activated it.

The following table presents the communication functions with synchronous application:

Function	Purpose...
SEND_TLG	sending a telegram.
RCV_TLG	receipt of a telegram.
CANCEL	stopping an exchange whilst in action.
ROR1_ARB	shifting a byte in a table to the right.
SWAP	swapping bytes in a table of words.

**Note:** Functions ROR1\_ARB and SWAP do not effect the communication processing, but they are necessary to handle the response of certain UNI-TE requests. For example, reading a table of words with the function SEND\_REQ .

### Availability of the functions according to their protocol

The following table shows the protocol supporting the communication functions:

Function	FIPWAY	FIPIO	Uni-telway	character mode	Modbus	Modbus Plus	TCP/IP ETHWAY
READ_VAR	X	X	X	-	X	X	X
WRITE_VAR	X	X	X	-	X	X	X
SEND_REQ	X	X	X	-	X	X	X
DATA_EXCH	X	X	X	-	-	-	X
PRINT_CHAR	X	-	-	X	-	-	X
INPUT_CHAR	X	-	-	X	-	-	X
OUT_IN_CHAR	X	-	-	X	-	-	X
SEND_TLG	X	-	-	-	-	-	-
RCV_TLG	X	-	-	-	-	-	-
READ_GDATA	-	-	-	-	-	X	-

Function	FIPWAY	FIPIO	Uni-telway	character mode	Modbus	Modbus Plus	TCP/IP ETHWAY
WRITE_GDATA	-	-	-	-	-	X	-
SERVER	-	-	-	-	X	-	-
WRITE_Asyn	-	-	-	-	-	-	TCP/IP
READ_Asyn	-	-	-	-	-	-	TCP/IP
<b>Caption :</b>							
X	Yes						
-	No						

## Target address

---

### Presentation

This parameter indicates the address of the target equipment of the exchange.

It can be localized:

- either by the internal words (%MW) or the internal constants (%KW),
- or written directly in immediate value.

To facilitate the preparation phase of the exchange, the PL7 language uses an operator with the syntax `ADR#` that authorizes the assignment of an immediate value, which is an address in a table that always has six internal words (%MW) or six constant consecutive words (%KW). See *Addressing*, p. 21 .

### Example

```
%MW1:6:=ADR#{2.4}\0.2.4\SYS
```

---

---

## Structure of management parameters

---

### Presentation

With four internal words (%MWk:4), parameters identify the address of the PL7 data used to monitor the communication functions.

The first two words %MWk and %MWK+1 are managed by the system.  
You will be in charge of managing the last two words %MWk+2 and %MWk+3.

### Structure

The four words %MWk:4 respect the following structure :

	Number of the word	Most significant byte	Least significant byte
Data managed by the system	%MWk	Number of exchange	Activity bit
	%MWk+1	Operation report	Communication report
Data managed by you	%MWk+2	Timeout	
	%MWk+3	Length	

### Activity bit

This bit corresponds to %MWk:x0. It signals the state of the application of the communication function.  
It is set at 1 whilst started and drops back down to 0 at the end of its execution.

### Number of exchange

During the sending of a communication function, the system automatically assigns it a number, allowing the identification of the exchange.  
This number allows the exchange in progress to be stopped if necessary (by the functions intermediaryCANCEL ).

## Management parameters: communication and operation reports

---

### Introduction

Communication and operation reports make up part of the management parameters.

**Note:** It is recommended that communication function reports be tested after use and before being started again. For cold starts, it is imperative that communication functions management parameters be reset to 0.

### Communication report

This report is common to all functions. It is significant when the value of the activity bit changes from 1 to 0.

The reports with a value between 16#01 and 16#FE concern the errors detected by the processor that has started the function.

The different values of this report are indicated in the following table:

Value	Communication report ( least significant byte )
16#00	Correct exchange
16#01	Stopping the exchange via a timeout
16#02	Stopping the exchange on user request CANCEL
16#03	Incorrect address format
16#04	Incorrect target address
16#05	Incorrect management parameter format
16#06	Incorrect specific parameters
16#07	Problem with sending to destination
16#08	Reserved
16#09	Size of reception buffer is insufficient
16#0A	Size of emission buffer is insufficient
16#0B	Absence of processor resource system
16#0C	Incorrect exchange number
16#0D	No telegram received
16#0E	Incorrect length
16#0F	Telegram service non configured
16#10	Network coupler absent
16#11	Request absence
16#12	Application server already active
16#13	UNI-TE V2 transition number incorrect



Value	Communication report ( least significant byte )
16#FF	Message refused

**Note:** The function can detect an error in the parameters before activating the exchange. In this case the activity bit remains 0, the report is initialized with the values corresponding to the default.

## Operation report

This report byte, which is specific to each function, specifies the effect of the operation on the remote application.

It is only significant if the communication report has the values:

- 16#00 (correct exchange),
- 16#FF ( message refused).

If the communication report has the value 16#00, the operation report has the following values:

Value	Communication report ( least significant octet )
16#00	Positive result
16#01	Non processed request
16#02	Incorrect response
16#03	Reserved

If the communication report has the value 16#FF, the operation report has the following values:

Value	Communication report ( least significant octet )
16#01	Lack of resource to the processor
16#02	Lack of line resource
16#03	Item of equipment absent or without resource (*)
16#04	Line error
16#05	Length error
16#06	Communication channel at fault
16#07	Addressing errors
16#08	Application error
16#0B	Absence of resource system
16#0C	Communication function non active
16#0D	Destination absent
16#0F	Intra-station routing problem or non configured channel

Value	Communication report ( least significant octet )
16#11	Address format not handled
16#12	Lack of destination resource
16#14	Non operational connection ( example: ETHERNET TCP/IP)
16#15	Lack of resource on the local channel
16#16	Non authorized access (example: ETHERNET TCP/IP)
16#17	Incoherent configuration network (example: ETHERNET TCP/IP)
16#18	Connection temporarily unavailable
16#21	Application server at a stop
<b>Caption :</b>	
(*)	Code uniquely managed by PCMCIA cards TSX FPP20 et TSX FPP10

---

## Management parameters : Length and Timeout

**Introduction** These two parameters are dependant upon you.

**Length** The length parameter is used both to specify the number of characters (in bytes) to send when sending a message and to store the number of characters (in bytes) received upon the receipt of a message.

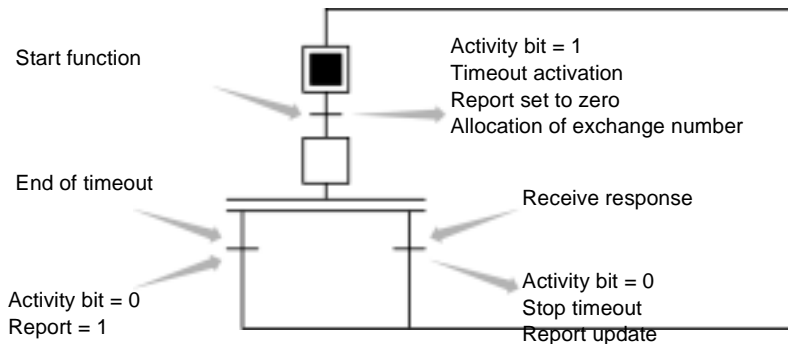
Each time before starting certain communication functions `SEND_REQ`, `DATA_EXCH`, `PRINT_CHAR`, `SEND_TLG` you must update the length parameter.

**Note:** For a function `PRINT_CHAR` for example, the length parameter `%MWk+3` must contain the number of octets (characters) to be sent before sending a message. After sending the characters, it contains the number of characters sent in the form of a communication report. If another function, which has a differing amount of bytes to be sent from the preceding function, uses the same report table during the application, you must initialize `%MWk+3` with the new number of bytes to be sent. If not `%MWk+3` maintains the number of bytes sent by the preceding function.

**Timeout** Timeout determines the maximum waiting time for a response. The time base for this parameter is 100 ms (value 0 corresponds to an infinite stand by value).

If timeout has elapsed, the exchange finishes with an error report, in the same way the receipt of a response is refused by the system after the end of timeout.

Example



**Note:** The timeout value of a communication function must be sufficient to guarantee the reception of the response to the question asked. This time depends on the type of network and on the load in effect at the moment of the transaction.

---

## Performances

---

### At a Glance

The exposed performances are as follows:

- the total number of communication functions executed by type of processor,
  - the communication function number by protocol and communication channel,
  - the maximum frame size.
- 

### Total number of communication functions

TSX Micro PLCs permit, at the most, simultaneous execution:

- of four communication functions with the terminal port (channel 0),
- of four communication functions with the PCMCIA card (channel 1).

TSX 57 10 PLCs permit, at the most, the simultaneous execution of 16 communication functions with all their communication channels.

TSX 57 20 PLCs permit, at the most, the simultaneous execution of 32 communication functions with all their communication channels.

TSX 57 25/30, PCX 57 35 and PMX 57 35 PLCs permit, at the most, the simultaneous execution of 48 communication functions with all their communication channels.

TSX 57 40/45 and PMX 57 45 PLCs authorize, at the most, the simultaneous execution of 64 communication functions with all their communication channels.

---

**Capacity of each communication channel**

The following table groups together the capacities of each communication channel simultaneously processing transactions according to different configurations.

Configuration	TSX Micro	TSX 57 10	TSX 57 20	TSX 57 25/30/40/45 PCX 57, PMX 57
Uni-telway master terminal port	4	4	4	4
Uni-telway master PCMCIA or SCY link	1	8	8	8
Uni-telway client slave terminal port	4	1	1	1
Uni-telway client slave PCMCIA or SCY link	1	1	1	1
Uni-telway server slave terminal port	4	4	4	4
Uni-telway server slave PCMCIA or SCY link	4	6	6	6
Modbus terminal port (1)	4	-	-	-
Modbus PCMCIA or SCY link	4	8	8	8
Character mode terminal block	1	1	1	1
Character mode PCMCIA or SCY link	4	8	8	8
FIPWAY messages	4	8	8	8
FIPWAY telegrams (2)	1 (10 ms)	1 (10 ms)	1 (10 ms)	1 (10 ms)
Modbus Plus	4	4	4	4
ETHERNET	-	16	16	16
<b>Key:</b>				
(-)	None			
(1)	Only for TSX Micro PLCs			
(2)	The TSX FPP 20 card permits the sending of telegrams every ten ms.			

**Note:** The performances of the communication functions of the TSX Micro and TSX Premium PLCs decrease when the MAST task period increases, but the cycle capacity remains constant.

**Note:** TSX Premium PLC terminal blocks only support 10 or 11 bit formats.

**Maximum frame size**

The maximum frame size depends on the communication channel, on the communication coupler and on the type of function executing the messaging.

Protocol	Channel	Maximum frame in bytes
Uni-telway	Terminal block	128
	TSX SCP 11• and TSX SCY 2160•	240
Character mode	Terminal block	120
	TSX SCP 11• and TSX SCY 2160•	4096
FIPWAY	TSX FFP 20	128
Modbus	TSX SCP 11• and TSX SCY 2160•	256
ETHWAY		256
TCP/IP		1024
Modbus Plus	TSX MBP 100	256

## Server function

---

### Introduction

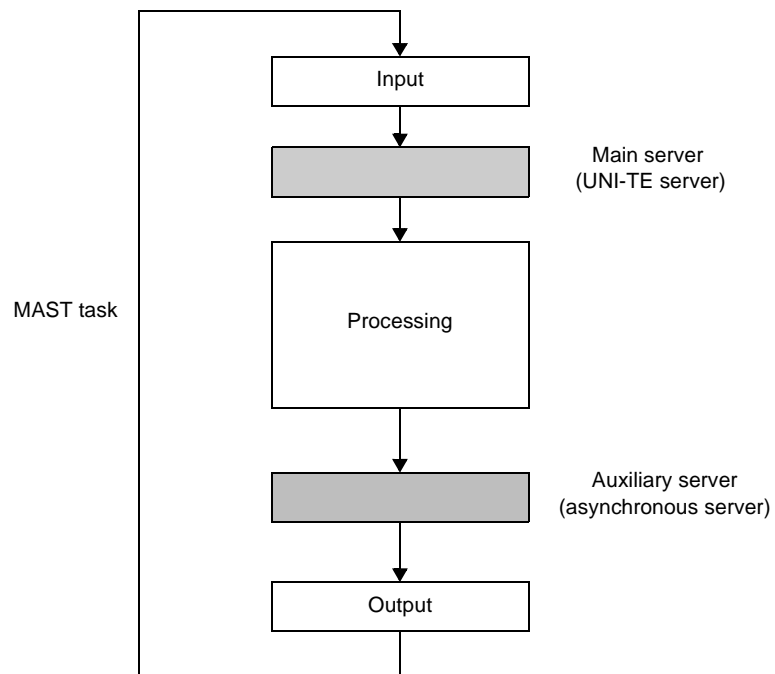
The server function allows response to requests from client equipment.

Processors TSX 57, 35, PCX 57 35, PMX 57 35 offer two request servers:

- a main server (recommended for requests with less than 256 bytes ),
  - a auxiliary server (recommended for requests with less than 1024 bytes),
- The two servers can be simultaneously activated.
- 

### Illustration

The following chart represents the prompting of the servers in the PLC cycle:





**Main server**

This server corresponds to port 0 (server UNI-TE). It is activated at the beginning of the PLC's MAST cycle.

The response time seen from the client PLC depends on the server PLC's cycle time. It allows the handling by PLC cycle of up to 4 simultaneous requests.

All UNI-TE requests are supported. The size of the request must be less than 256 bytes.

This item is addressable to the topological address SYS or {network.station}SYS.

---

**Subordinate server**

This server corresponds to port 7 (server UNI-TE). It is only activated as a periodic task at the end of the PLC cycle after the processing of the MAST task, whilst waiting for the beginning of the following cycle.

More importantly the beginning of the following cycle could interrupt a request that is in action. Access to this server is therefore reserved to applications that do not require any form of coherence in the read or written data.

The application response time will be essentially a function of the PLC cycle time.

The size of the response can be 1024 bytes. It is not accessible via a communication function, the server handled the requests READ/WRITE object (bit or word), Read object list...

---

## 3.2 Help whilst entering the communication functions

---

### Presentation

#### Subject of sub-chapter

This sub-chapter presents help when entering communication functions with the software PL7.

---

#### What's in this section?

This section contains the following topics:

Topic	Page
Help with entering the communication functions	67
Accessing a specific function, method or procedure type instruction	68
Help on address input	70

---

## Help with entering the communication functions

### Introduction

Whilst programming you can have access to the input help screen giving information about all the parameters of a communication function.

This help is accessed via the functions in the PL7 software library.

### Illustration

The following illustration shows the help screen when entering a communication function.

**Note:** The number and the type of field vary according to the communication function selected.

### Availability

This screen is available for the following communication functions:

- DATA\_EXCH
- INPUT\_CHAR
- OUT\_IN\_CHAR
- PRINT\_CHAR
- READ\_VAR
- SEND\_REQ
- SEND\_TLG
- WRITE\_VAR

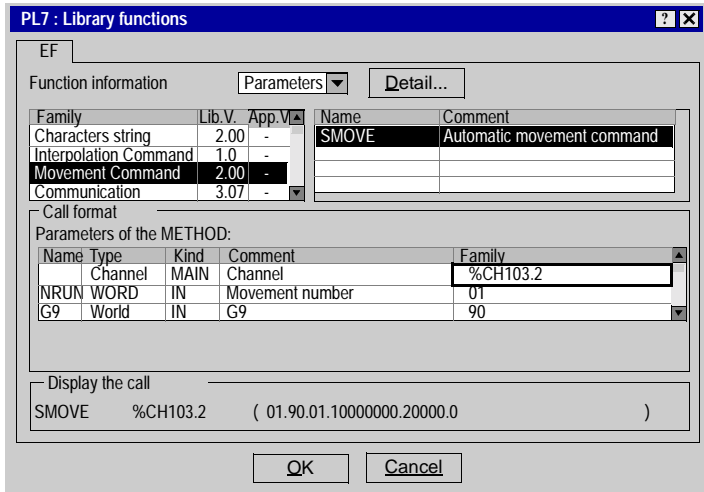
## Accessing a specific function, method or procedure type instruction

### Presentation

Application-specific function entry can be accessed by:

- directly entering the instruction and its parameters in an operate block,
- by the entry help function accessible through the program editors (LD, IL, ST).

### Calling up a function

Step	Action
1	Access the required editor.
2	<p>Depending on the editor used, choose one of the following methods to open the function library.</p> <ul style="list-style-type: none"> <li>• Press <b>Shift + F8</b> (LD, IL, ST editors).</li> <li>• Click on the ((LD editor) icon.</li> <li>• Select the command <b>Utilities</b> → <b>Enter call for a</b> ((IL, ST editors) function.</li> </ul> <p><b>Note:</b> The function library appears.</p> 
3	Select the specific application in the <b>Family</b> field.
4	Select the instruction in the <b>Name</b> field.
5	Many of the instructions have a customized entry help screen. Access this screen by clicking on the <b>Details</b> button.

Step	Action
6	Enter each instruction parameter (each instruction is developed in the relevant application-specific documentation) <ul style="list-style-type: none"><li>● in the customized screen</li><li>or</li><li>● in the <b>Entry field</b> in the <b>Library Functions</b> screen. To do this, the <b>Parameter</b> item must be selected in the <b>Function Information</b> field.</li></ul>
7	Click <b>Ok</b> to confirm.

---

## Help on address input

### Introduction


To facilitate the input of the address, a help screen is given.

This screen allows a description of the architecture in which the communication function is integrated and generated.

In completing the fields of this description, the address is automatically generated.

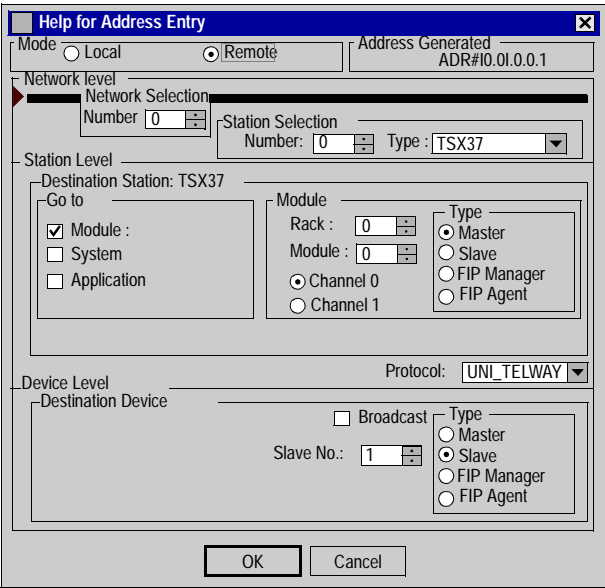
### How to access the help

When entering the parameters of the communication function, you can access help on the address entry in the following way:

Step	Action
1	Select the following button: 

### Illustration

The following illustration shows the Help screen when entering the address for a communication function.



---

<b>Mode</b>	<p>The first parameter to select is the <b>Mode</b>. It allows the selection of a type of communication:</p> <ul style="list-style-type: none"><li>• local (bus communication),</li><li>• remote (network communication).</li></ul>
<b>Network level</b>	<p>Only for remote communications, <b>network level</b> allows:</p> <ul style="list-style-type: none"><li>• input of the network number,</li><li>• input of the station number,</li><li>• selection of the type of station.</li></ul>
<b>Station level</b>	<p>This parameter allows according to the communication function to select the type of exchange:</p> <ul style="list-style-type: none"><li>• The box <b>Application</b> selects an exchange with a PL7 application (corresponds to APP addressing).</li><li>• The box <b>System</b> selects the PLC system of the station designated by the network level (corresponds to the address SYS).</li><li>• The box <b>Module</b> signifies that the destination device is connected to the station by a link (Uni-telway, Modbus, Modbus Plus ou FIPIO). This case requires the information:<ul style="list-style-type: none"><li>• the position of the module which supports this link,</li><li>• the type of this module.</li></ul></li></ul>
<b>Protocol</b>	<p>The field <b>Protocol</b> defines the exchange protocol between the network station and the exchange destination device.</p>
<b>Device Level</b>	<p>This parameter allows you to specify:</p> <ul style="list-style-type: none"><li>• the type of destination device,</li><li>• the address of this device.</li></ul>
<b>Limitations</b>	<p>In the Help screen for address entry, certain communications (from a Uni-telway slave) make it necessary to code the address of the destination in the send buffer. See Transmitting UNI-TE Requests: SEND_REQ, p. 95.</p> <p>The Help window allows complete input of the part corresponding to ADR# while warning the user that he must code the supplementary buffer.</p> <p>The coding of remote station addresses is only supported by the following devices: TSX 17, TSX 37, TSX 47-107, TSX 57.</p> <p>For third party devices only the input of the port number is proposed, in other cases the input of the address must be made manually.</p>

---

### 3.3 Description of the different communication functions

---

#### Presentation

---

#### Subject of sub- chapter

This sub-chapter presents every type of communication function.

---

#### What's in this section?

This section contains the following topics:

Topic	Page
Reading standard objects: READ_VAR	74
Reading standard objects : input Help screen	77
Reading standard objects : example of use	78
Reading standard objects : example of network use	79
Reading standard objects : example of exchange of variables with parameter monitoring	81
Reading standard objects : More detail on reading bits	83
Reading standard objects : reading a timer's current parameters	85
Reading standard objects : reading the current parameters of a monostable	86
Writing standard objects : WRITE_VAR	87
Writing Standard Objects: Entry screen help	89
Writing standard objects: example of use	90
Writing standard objects: example of network use	91
Writing standard objects: example of exchange of variables with parameter monitoring	93
Transmitting UNI-TE Requests: SEND_REQ	95
UNI-TE request transmission input Help screen	97
UNI-TE request transmission example of network use	98
UNI-TE request transmission: List of requests	100
Exchange of text data : DATA_EXCH	104
Exchange of text data : input Help screen	106
Exchange of text data : examples of use	107
Text Type Data Exchange: examples of use with an Altivar	109
Sending a telegram : SEND_TLG	111
Sending a telegram: entry help screen	112



Topic	Page
Sending a telegram : example of use	113
Receiving a telegram : RCV_TLG	114
Receiving a telegram : example of use	116
Writing a string of characters. PRINT_CHAR	117
Writing a chain of characters : input help screen	119
Writing of a chain of characters : example of use	120
Reading a character string: INPUT_CHAR	122
Reading a string of characters : input Help screen	124
Reading a character string : example of use	125
Sending/receiving a string of characters : OUT_IN_CHAR	126
Sending/receiving a string of characters : input Help screen	128
Sending/receiving a string of characters : example of use	129
Stopping an exchange whilst in action. : CANCEL	131
Stopping an exchange whilst in action : example of use	132
Shifting a byte to the right in a table : ROR1_ARB	134
Shifting a byte to the right in a table : example of use	135
Swapping bytes in a word table : SWAP	137
Reading of common Modbus Plus data: READ_GDATA	138
Writing of Modbus Plus common data : WRITE_GDATA	139
Immediate server : SERVER	140
Immediate server : example of use	142
Asynchronous messaging service : WRITE_Asyn and READ_Asyn	143

## Reading standard objects: READ\_VAR

---

### Introduction

The function `READ_VAR` allows reading of the value of one or several language objects:

- internal bit, internal word, system bit, system word, constant word, double internal word, double constant word,
- structured objects (timer, monostable, counter, register, drum).

The objects read must always be consecutive. They can be located in a remote CPU or in a device connected by a communication channel of the type ETHERNET TCP/IP, FIPWAY, Uni-telway, terminal port, Modbus, Modbus Plus, Modem.

The reply must contain a maximum number of bytes depending on the protocol and the type of destination product. See Performances, p. 61.

At the end of the reading operation, the length of the received data is stored in word 4 of the management parameter. See Structure of management parameters, p. 55.

The function `READ_VAR` can read up to 1000 consecutive bits in a remote device, whatever this device is and whatever protocol is used (Uni-telway ou Modbus/Jbus).

<p><b>Note:</b> Reading more than 1000 bits requires the use of the function <code>SEND_REQ</code>. Note that PLCs TSX 07, TSX 37, TSX 57 cannot send more than 1000 bits following a read request.</p>
---

**Syntax**

The syntax of the communication function `READ_VAR` is presented in the following form:

```
READ_VAR(ADR#0.0.6, '%MW', 100, 10, %MW10:10, %MW40:4)
```

The following table shows the different function parameters.

Parameter	Description
ADR#0.0.6	Address of the exchange destination. The following addresses {Network.Station}APP, {Network.Station} APP.num and addresses in circulation are prohibited in this field.
'%MW'	String of characters specifying the type of object to be read: <ul style="list-style-type: none"> <li>● %I: external input bit</li> <li>● %IW: external input word</li> <li>● %M: internal bit</li> <li>● %MW: internal word</li> <li>● %S: system bit</li> <li>● %SW: system word</li> <li>● %KW: constant word</li> <li>● %MD: double internal word</li> <li>● %KD: double constant word</li> <li>● %T: timer (PL7-3)</li> <li>● %TM: timer (standard CEI 1131)</li> <li>● %MN: monostable</li> <li>● %R: register</li> <li>● %C: counter</li> <li>● %DR: drum</li> </ul>
100	Double word indicating index of the first object to be read.
10	Word specifying the number of objects to be read.
%MW10:10	Table of words containing the value of objects read.
%MW40:4	Management parameters. The operation report takes one of the following values: <ul style="list-style-type: none"> <li>● 16#00: correct read operation</li> <li>● 16#01: operating error</li> <li>● 16#02: incorrect response</li> <li>● 16#03: size of the response is inconsistent</li> </ul>

**Note:** It is not necessary to initialize the length parameter before starting the function.

In Modbus, only objects %M and %MW are usable.

## Rules for use

In PL7, the input of object types must be consistent. Writing must be either all in lower case or all in upper case, if not the function sends a report of 16#06 (incorrect specific parameters).

For the PLCs TSX Micro or TSX Premium, read access of internal bits (via Unitelway) has the following feature:

- the forcing value of the bits is returned in the reponse.
- the reading of 1 bit thus comprises two response bytes:
  - the first contains the value of the 8 bits from that requested.
  - the second contains the identifier of the forcing of these bits.
- to read one of the last 8 bits from the memory it is obligatory to read the last 8, if not the function sends a operating report 16#01.

<p><b>Note:</b> Preview in the size of the reception table, the forcing identifier bytes. If not, error code 16#03 is returned in the report.</p>
---

---

## Reading standard objects : input Help screen

### Introduction

For this communication function, you can call up the input help screen.

### Function parameters

This function supports six parameters:

Parameters	Types of objects	Comments
Address	ADR# %MWX:n	If you input a value directly into the field, the Help button for the inputting of addresses is greyed out.
Type of object to be read	%MW	An object list is suggested to you.
Address of first object to be read	%MDx %KDx Immediate value	In %KDx, an input field for the value is displayed.
Number of consecutive objects	%MWx %KWx immediate value	In %KWx, a value input field is displayed.
Response	%MWx:n	-
Report	%MWx:4	-

**Note:** The symbols are accepted.

### Example

The following screen shows an example of function entry:

**READ\_VAR**

Parameters

Address : ADR#0.0.6

Type of object to Read: %MW

Address of first object to Read: %KD16 100

Number of consecutive Objects to Read: 10

Reception Zone: %MW10 10

Report: %MW40 4

Possible types : %MWin (n=4)

OK Cancel

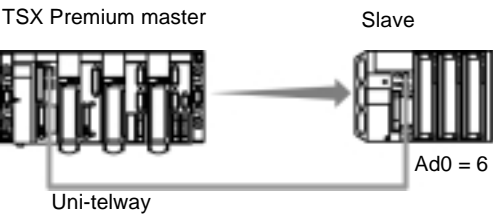
## Reading standard objects : example of use

### Introduction

The example deals with the master station reading the internal words %MW100 à %MW109 from station with address 6 on a Uni-telway bus. The value of the words read must range from the internal word %MW10, the management parameters are introduced from %MW40.

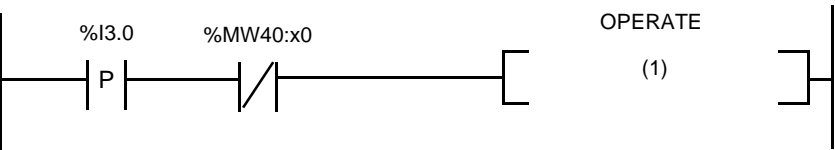
### Illustration

The two stations are connected by Uni-telway bus.



### Send

Programming of the function is as follows:



(1) READ\_VAR(ADR#0.0.6, '%MW', 100, 10, %MW10:10, %MW40:4)

Query parameters:

Parameters	Description
ADR#0.0.6	<ul style="list-style-type: none"><li>● 0: module</li><li>● 0: channel 0</li><li>● 6: sender address Ad0</li></ul>
'%MW'	Type of object (internal word)
100	address of first object
10	Number of consecutive objects
%MW10:10	Content of reply
%MW40:4	Report

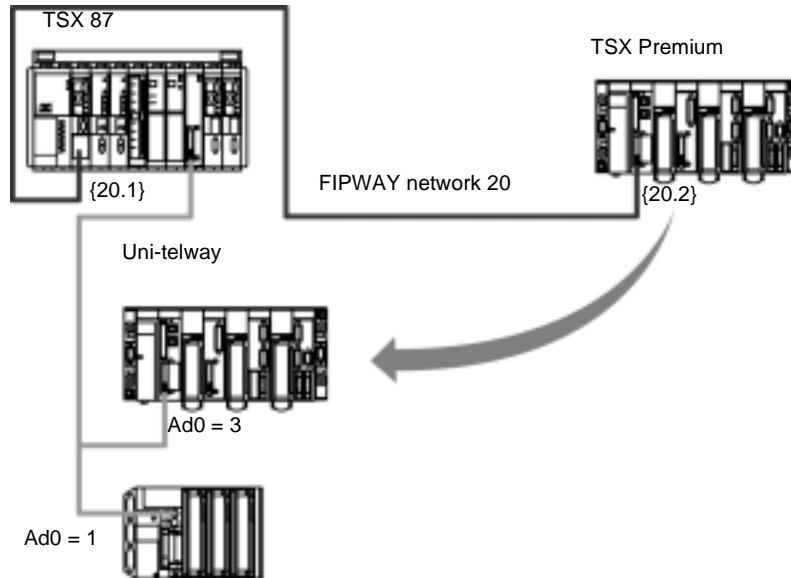
## Reading standard objects : example of network use

### Introduction

The example deals with the reading of a table of 5 words %MW0 à %MW4 from UNI-telway slave with network address 20, station 1, communication coupler TSX SCM 2116 in slot 5, channel in the communication coupler 1, server address Ad0 = 3.

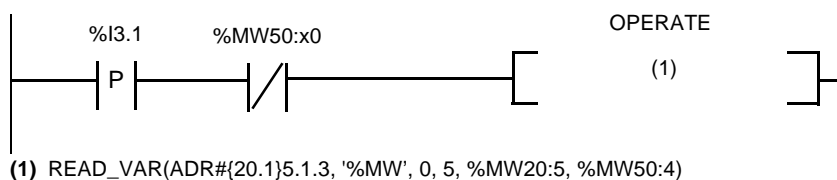
### Illustration

The two stations are connected via a FIPWAY network.



### Send

Programming of the function is as follows:



Query parameters:

Parameters	Description
ADR#{20.1}5.1.3	<ul style="list-style-type: none"><li>● {20.1}: network 20, station 1</li><li>● 5: module</li><li>● 1: channel 1</li><li>● 3: sender address Ad0</li></ul>
'%MW'	Type of object (internal word)
0	address of first object
5	Number of consecutive objects
%MW20:5	Content of reply
%MW50:4	Report

---



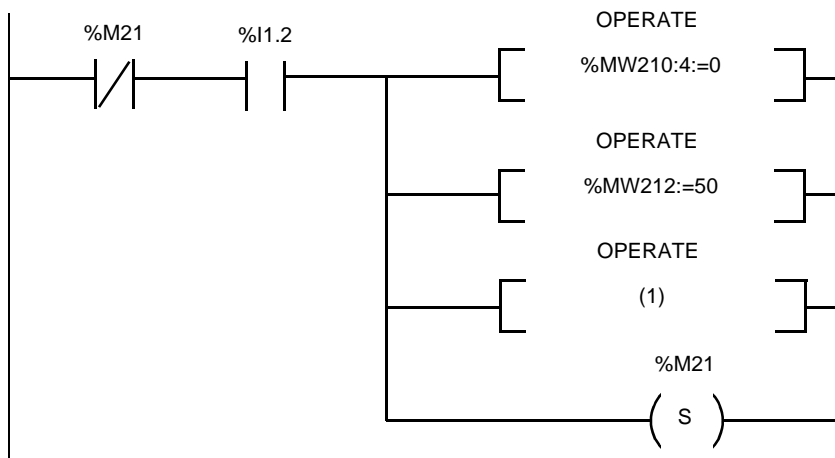
## Reading standard objects : example of exchange of variables with parameter monitoring

### Introduction

The following example illustrates the communication function `READ_VAR` with management parameter monitoring.

### Programming the function

Reading a variable

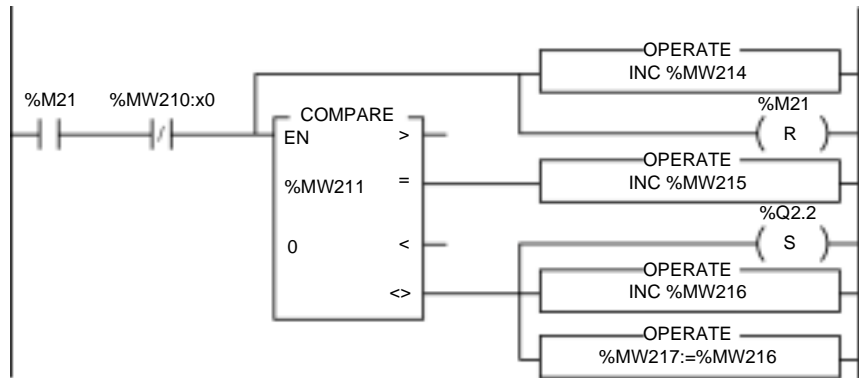


**(1)** `READ_VAR(ADR#3.1.7, '%MW', 20, 1, %MW1701:1, %MW210:4)`

- input bit %I1.2 allows a function to be activated.
- the internal bit %M21 allows the activity of the function to be tested.
- the first block `OPERATE` allows initialization of the management parameters.
- the second block `OPERATE` allows the value of the timeout to be pre-set at 5s.

**Programming  
the monitoring**

## Exchange monitoring



- the internal word %MW214 counts the number of exchanges.
- the internal word %MW215 counts the number of correct exchanges.
- the internal word %MW216 counts the number of failed exchanges.
- the internal word %MW217 stores the error message.
- the external output bit %Q2.2 allows external warning of an exchange error.

## Reading standard objects : More detail on reading bits

### Introduction

The following examples illustrate the communication function `READ_VAR` for the reading of bits.

### Reading internal 32 bits

The function syntax for reading internal bits is as follows:

```
READ_VAR(ADR#{20.1}5.1.3, '%M', 0, 32, %MW100:4, %MW50:4)
```

The reception table must consist of 8 bytes (4 words), 4 bytes for the value and 4 bytes for the forcing identifier.

Value	%MW100	0000	0000	1100	1100
	%MW101	1111	1111	0000	1111
Forcing	%MW102	0000	0000	0101	0101
	%MW103	0000	0000	0000	1111

There is forcing when a flag is set to 1, the forcing value is that of the corresponding read bit.

### Example

	Value of the bits	Bits forcing	Description
Byte 0	0	1	The bit is forced with the value 0.
	0	0	The bit is not forced.
	1	1	The bit is forced with the value 1.
	1	0	The bit is not forced.
	0	1	The bit is forced with the value 0.
	0	0	The bit is not forced.
	1	1	The bit is forced with the value 1.
	1	0	The bit is not forced.

**Reading internal  
16 bits**

The function syntax for reading internal bits is as follows:  
`READ_VAR(ADR#{20.1}5.1.3, '%M', 0, 18, %MW100:3, %MW50:4)`

The reception table must contain 3 words (or 6 bytes). In effect, to obtain the value of 18 bits, 3 bytes are necessary (modulo 8 nearest to 18) and 3 bytes in addition to contain the forcing value of the 18 bits.

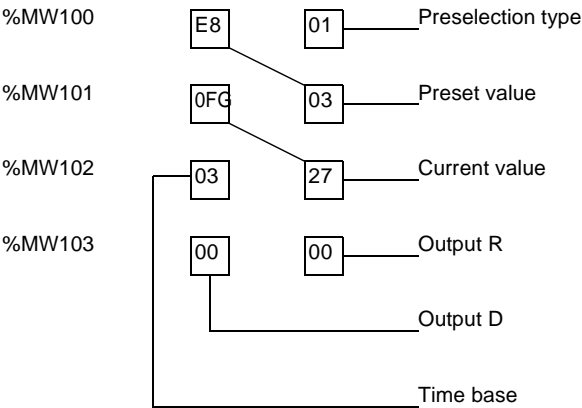
Value	%MW100	0000	0000	1111	1111
Forcing	%MW101	0000	0000	0000	1111
	%MW102	0000	1111	0000	0000

## Reading standard objects : reading a timer's current parameters

**Introduction** The following example illustrates the communication function `READ_VAR` to read the parameters of a timer `%T` or `%TM`.

**Reading of a timer `%T`** The function syntax for reading a timer is as follows:  
`READ_VAR(ADR#{17.1}7.1.3, '%M', 0, 1, %MW100:4, %MW50:4)`

The reception table must contain 8 bytes (4 words).



**Interpretation of the read bytes** The bytes are interpreted in the following manner:

Value	Comments
01	Type of preset (see reference manual)
03 E8	Preset value (1000)
27 0F	Current value of timer (9999)
03	Time base (1min.)
00	Current value of the output R (0 or 1)
00	Current value of the output D (0 or 1)

## Reading standard objects : reading the current parameters of a monostable

### Introduction

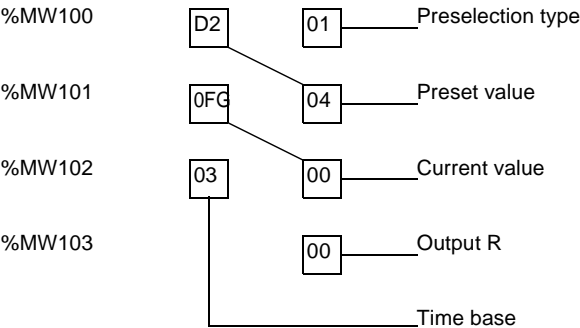
The following example illustrates the communication function `READ_VAR` to read the parameters of a monostable `%MN`.

### Reading a monostable

The function syntax for reading a timer is as follows:

```
READ_VAR(ADR#{17.1}7.1.3, '%MN', 0, 1, %MW100:4, %MW50:4)
```

The reception table must contain 7 bytes (4 words).



### Rules for use

If you read the current parameters of two monostables, the parameters of the second begin at the end of the end of the parameters of the first from byte 8 onwards. There is therefore a shift of one byte which must be taken into account when interpreting the read parameters.

The case is identical when reading register parameters which are read on an odd number of bytes (9).

In this case, if you read the parameters of several consecutive functions, there will be shift of 1 byte to the left for even parameters in relation to odd parameters. This shift does not take place when the number of bytes in the read parameters is even.

## Writing standard objects : WRITE\_VAR

### Introduction

The function `WRITE_VAR` allows the writing of the value or values of one or several language objects of the same type (internal bit, internal word, constant word, system bit, system word, double internal word, double constant word).

The objects to be written must always be consecutive. They can be located in a remote CPU or in a device connected by a communication channel of the type Uni-telway, terminal port, FIPWAY, Modbus, Modbus Plus, ETHWAY.

### Syntax

The syntax of the communication function `WRITE_VAR` is presented in the following form:

```
WRITE_VAR(ADR#0.0.6, '%MW', 0, 50, %MW100:50, %MW50:4)
```

The following table describes the different parameters of the function.

Parameter	Description
ADR#0.0.6	Address of the exchange destination. The following addresses {Network.Station}APP, {Network.Station}APP.num and addresses in circulation are prohibited in this field.
'%MW'	String of characters specifying the type of object to be read: <ul style="list-style-type: none"> <li>● %M: internal bit</li> <li>● %MW: internal word</li> <li>● %S: system bit</li> <li>● %SW: system word</li> <li>● %KW: constant word</li> <li>● %MD: double internal word</li> <li>● %KD: double constant word</li> <li>● %T: timer (PL7-3)</li> <li>● %TM: timer (standard IEC 1131)</li> <li>● %MN: monostable</li> <li>● %R: register</li> <li>● %C: counter</li> <li>● %DR: drum</li> </ul>
0	Double word indicating index of the first object to be written.
50	Word specifying the number of objects to be written.
%MW100:50	Table of words containing the value of objects to be sent.
%MW50:4	Management parameters. The operation report takes one of the following values: <ul style="list-style-type: none"> <li>● 16#00: correct read operation</li> <li>● 16#01: operating error</li> <li>● 16#02: incorrect response</li> </ul>

**Note:** The length parameter must not be initialized before the start of the function.  
In Modbus, only objects %M and %MW are usable.

---



## Writing Standard Objects: Entry screen help

**At a Glance**      The Entry Help Screen may be consulted in this communication function.

**Parameters of the Function**      This function supports six parameters:

Parameters	Type of Object	Comments
Address	ADR# %MWX:n	If you enter a value directly into the field, the Entry Help button is grayed out.
Type of Object to Write	%MW	You are offered a list of objects.
Address of first object to write	%MDx %KDx Immediate Value	In %KDx, an entry field for the value is displayed.
Number of consecutive objects to write	%MWx %KWx Immediate Value	In %KWx, an entry field for the value is displayed.
Data to Write	%MWx:n	-
Report	%MWx:4	-

**Note:** Symbols are accepted.

**Example**      The following screen shows an example of function entry:

WRITE\_VAR

Parameters

Address:

ADR#0.0.6

Type of Object to Write:

%MW

Address of first object to write:

%KD16

Number of consecutive objects to write :

10

Data to Write:

%MW10

10

Report:

%MW40

4

Possible types: %KWin, %MWin, (n=6)

Immediate Address (ADR#)

OK

Cancel

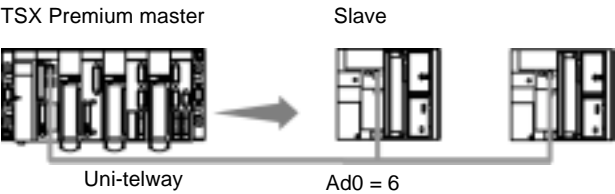
## Writing standard objects: example of use

### Introduction

The example shows the writing by the master station of 50 internal words %MW0 à %MW4 to the address slave 6 (Ad0=6). The value of the words to be written is found in the internal words %MW100 à %MW149 of the master station, the management parameters are resident from %MW50.

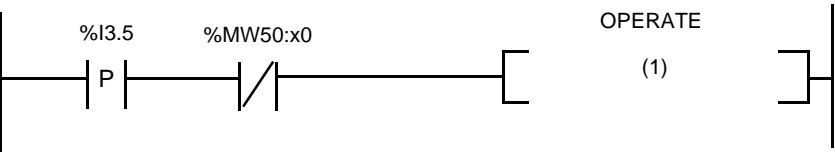
### Illustration

The two stations are connected by Uni-telway bus.



### Send

Programming of the function is as follows:



(1) WRITE\_VAR(ADR#0.0.6, '%MW', 0, 50, %MW100:50, %MW50:4)

Query parameters:

Parameters	Description
ADR#0.0.6	<ul style="list-style-type: none"><li>0: module</li><li>0: channel 0</li><li>6: sender address Ad0</li></ul>
'%MW'	Type of object (internal word)
0	address of first object
50	Number of consecutive objects
%MW100:50	Content of reply
%MW50:4	Report

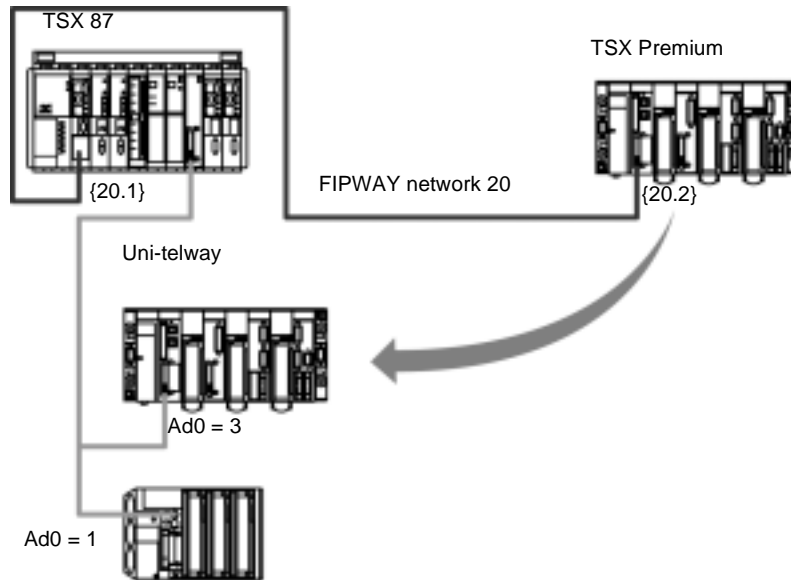
## Writing standard objects: example of network use

### Introduction

The example shows the writing of a 50-word table %MW0 à %MW49 to the slave Uni-telway network address 20, station 1, communication coupler TSX SCM 2116, slot 5, communication coupler channel 1, server address Ad0 = 3. The values to be written are in the words %MW0 à %MW49 of the transmitter, the management parameters reside from %MW 100.

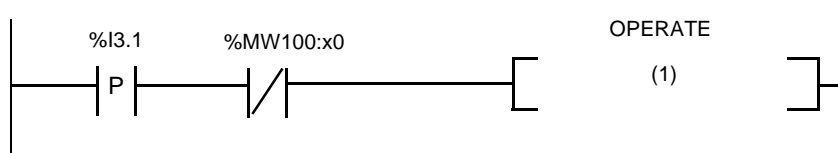
### Illustration

The two stations are connected via a FIPWAY network.



### Send

Programming of the function is as follows:



(1) WRITE\_VAR(ADR#{20.1}5.1.3, '%MW', 0, 50, %MW0:50, %MW100:4)

Query parameters:

Parameters	Description
ADR#{20.1}5.1.3	<ul style="list-style-type: none"><li>● {20.1}: network 20, station 1</li><li>● 5: module</li><li>● 1: channel 1</li><li>● 3: server address Ad0</li></ul>
'%MW'	Type of object (internal word)
0	address of first object
50	Number of consecutive objects
%MW0:50	Datas to be written
%MW100:4	Report

---

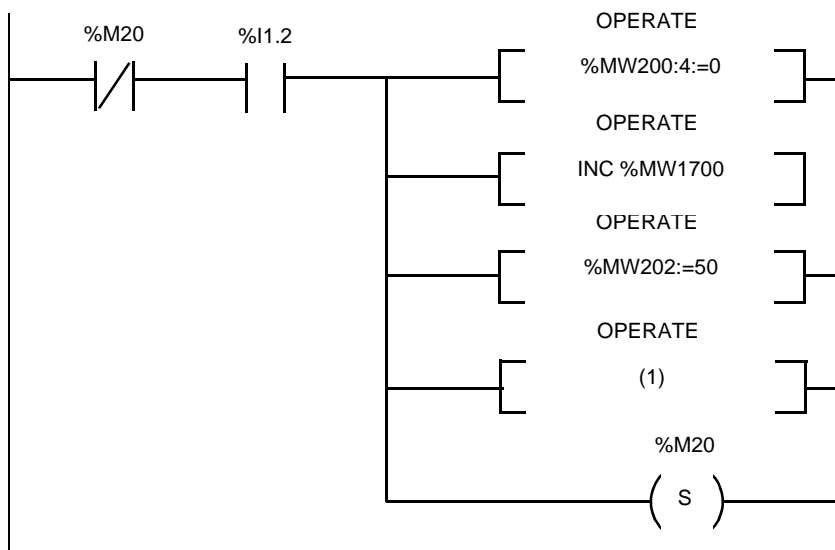
## Writing standard objects: example of exchange of variables with parameter monitoring

### Introduction

The following example illustrates the communication function `WRITE_VAR` with management parameter monitoring.

### Programming the function

Writing a variable

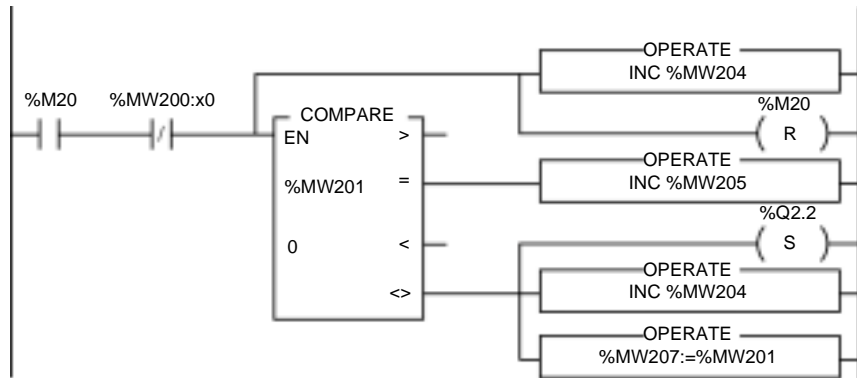


**(1)** `WRITE_VAR(ADR#3.1.7, '%MW', 20, 1, %MW1700:1, %MW200:4)`

- Input bit `%I1.2` allows a function to be activated.
- The internal bit `%M20` allows the activity of the function to be tested.
- The first block `OPERATE` allows initialization for the management parameters.
- The second block `OPERATE` allows the value of the word `%MW1700` to be incremented.
- The third block `OPERATE` allows the value of the timeout to be pre-set at 5s.

## Programming the monitoring

### Exchange monitoring



- The internal word %MW204 counts the number of exchanges.
- The internal word %MW205 counts the number of correct exchanges.
- The internal word %MW206 counts the number of failed exchanges.
- The internal word %MW207 stores the error message.
- The external output bit %Q2.2 allows external warning of an exchange error.

## Transmitting UNI-TE Requests: SEND\_REQ

### At a Glance

The `SEND_REQ` function allows the coding and transmission of all UNI-TE requests and Modbus/Jbus as well as the associated response reception.

**Note:** In some cases it is necessary to reclassify the received objets through the `SEND_REQ` function (see Shifting a byte to the right in a table : `ROR1_ARB`, p. 134).

Details of the UNI-TE requests coding are provided in the TSX DR NET Communication Reference Manual document, the coding of Modbus/Jbus requests is provided in the TSX DG MDB manual.

### Syntax

The `SEND_REQ` communication function syntax appears in the following form:

```
SEND_REQ (ADR#0.0.6, 15, %MW0:1, %MW150:24, %MW40:4)
```

The following table describes the function's different parameters.

Parameter	Description
ADR#0.0.6	The destination entity address for the exchange. The following {Station Network} APP addresses, {Station Network}APP.num and the (ALL) broadcast addresses are forbidden in this field.
15	This parameter specifies the value of the request code in accordance with the UNI-TE standard. It is decimalized by default (example: 252 for non-solicited data requests). If the user wants to code the requests hexadecimally, the code request must have the following syntax: 16# followed by a hexadecimal code request (example 16#FC for the request for non-solicited data).
%MW0:1	Data to be transmitted. The size of the word table depends upon the send request. It must have a minimum length of one word even if the request does not consist of any particular data to be transmitted (request Run, Stop, Identification, etc.). The length of the data to be transmitted must be memorized in the fourth word of the parameter manager (word length) before launching this function.
%MW150:24	Word table containing the reply data. It must have a minimum length of one word even if the request does not consist of any particular data to be received (request non-solicited data). The length of the data actually received is indicated at the end of the exchange, in the fourth word of the parameter manager.

Parameter	Description
%MW40 : 4	<p>Parameter Manager The operation report takes one of the following values:</p> <ul style="list-style-type: none"><li>● 16#00: correct operation</li><li>● 16#02: incorrect reply</li><li>● 16#FD: operational error</li><li>● Other values:<ul style="list-style-type: none"><li>● Request code + 16#30: upon positive reply for certain requests</li><li>● 16#FE: upon positive reply for certain requests</li><li>● 16#FB: upon reply to a mirror request</li></ul></li></ul>

**Note:** The fourth word of the parameter manager table corresponds to the length parameter. Before the function execution it must contain the length (in bytes) of the data that is to be transmitted. Once the request has been carried out it is automatically updated and contains the length of the reply (in bytes).  
The `SEND_REQ` function allows more than 1000 bits to be read in a remote device. Note that the TSX 07, TSX 37, TSX 57 PLCs can return no more than 1000 bits following a read request.

---



## UNI-TE request transmission input Help screen

### Introduction

For this communication function, you can call up the input Help screen.

### Function parameters

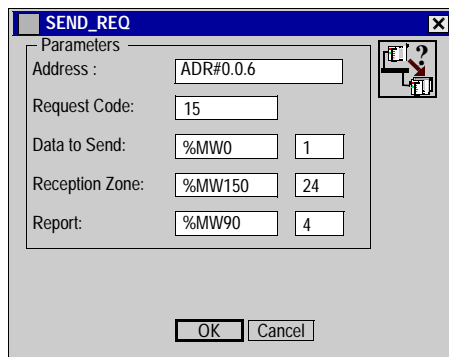
This function supports five parameters:

Parameters	Types of objects	Comments
Address	ADR# %MWX:n	If you input a value directly into the field, the Help button for the inputting of addresses is greyed out.
Request code	%MWx %KWx immediate value	In %KWx, a value input field is displayed.
Data to be sent.	%MWx:n %KWx:n	-
Response	%MWx:n	-
Report	%MWx:4	-

**Note:** The symbols are accepted.

### Example

The following screen shows an example of inputting function input:



The screenshot shows a dialog box titled "SEND\_REQ" with a "Parameters" section. It contains five rows of input fields:

- Address : ADR#0.0.6
- Request Code: 15
- Data to Send: %MW0 1
- Reception Zone: %MW150 24
- Report: %MW90 4

At the bottom are "OK" and "Cancel" buttons. A help icon (question mark) is visible in the top right corner of the dialog box.

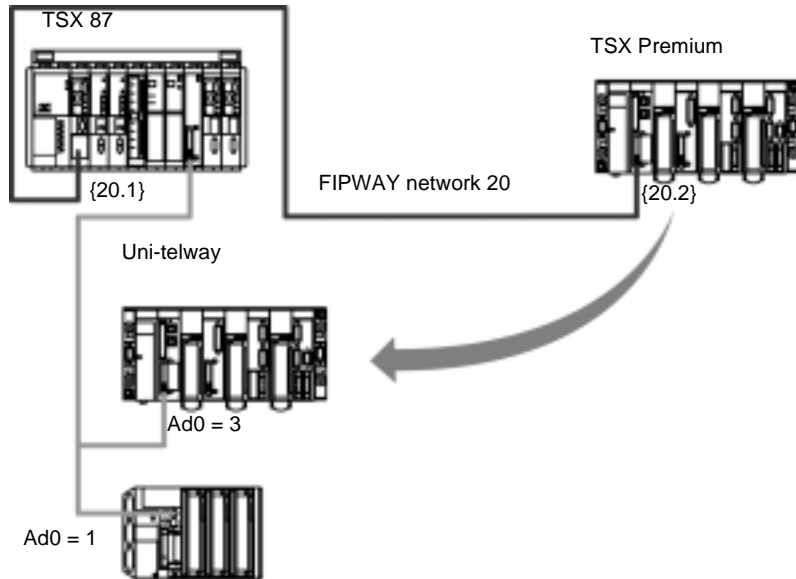
## UNI-TE request transmission example of network use

### Introduction

The example shows the identification by the master station {Network 20. Station 2} of the station connected on network 20 station 1 and address Ad0 = 3 on Uni-telway. The identification request has the code 15 in decimal (or 0F in hexadecimal).

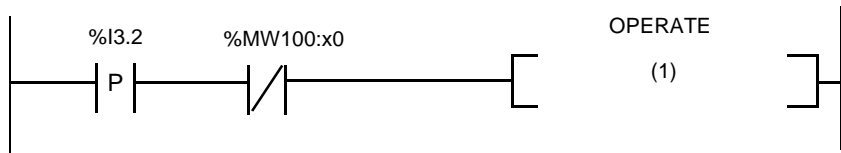
### Illustration

The two stations are connected via a FIPWAY network.



### Send

Programming of the function is as follows:



(1) SEND\_REQ(ADR#{20.1}5.1.3, 15, %MW0:1, %MW150:24, %MW100:4)

Query parameters:

Parameters	Description
ADR#{20.1}5.1.3	<ul style="list-style-type: none"><li>● {20.1}: network 20, station 1</li><li>● 5: module</li><li>● 1: channel 1</li><li>● 3: server address Ad0</li></ul>
15	Request 15 (or 16#0F if coding is in hexadecimal)
%MW0:1	Data sent (for the example: no data to send)
%MW150:24	Contents of the response (receipt of 24 words)
%MW100:4	Report

**Note:** Before starting the function each time, initialize the length parameter (in the example: %MW103 = 0).

## UNI-TE request transmission: List of requests

---

### Introduction

UNI-TE protocol allows:

- identification and diagnosis of all equipment using a UNI-TE server.
- provision of a set of services for read/write access to data types.
- to download data between devices.
- protection of a server to avoid conflicting access during a critical period.

These different services are accessed by the `SEND_REQ` with a code for the UNI-TE request to be sent.

---

### General use requests

These requests allow the identification and diagnosis of all types of equipment using a UNI-TE server.

Name of request	Code request	Code report	Comment
IDENTIFICATION	16#0F	16#3F	provides information <ul style="list-style-type: none"><li>• product range,</li><li>• specific-application type,</li><li>• product type,</li><li>• catalog reference.</li></ul>
READ_CPU	16#4F	16#7F	performs system diagnostics on all equipment.
PROTOCOLE_VERSION	16#30	16#60	allows adaptation of the protocol version for two communicating items.
MIRROR	16#FA	16#FB	tests for tracking of information between 2 communicating devices.

---

### Access to standard objects

These requests provide a set of services allowing read/write access to bit and internal word data, word and bit systems, floating points, constants and Grafcet data.

Name of request	Code request	Code report	Comment
READ_INTERNAL_BIT	16#00	16#30	allows the value of an internal bit to be read.
WRITE_INTERNAL_BIT	16#10	16#FE	allows the value of an internal bit to be written.
READ_SYSTEM_BIT	16#01	16#31	allows the value of a system bit to be read.
WRITE_SYSTEM_BIT	16#11	16#FE	allows the value of a system bit to be written.
READ_INTERNAL_WORD	16#04	16#34	allows the value of an internal word to be read.
WRITE_INTERNAL_WORD	16#14	16#FE	allows the value of an internal word to be written.
READ_SYSTEM_WORD	16#06	16#36	allows the value of a system word to be read.

---

Name of request	Code request	Code report	Comment
WRITE_SYSTEM_WORD	16#15	16#FE	allows the value of a system word to be written.
READ_CONSTANT_WORD	16#05	16#35	allows the value of a constant word to be read.
FORCE_INTERNAL_BIT	16#1B	16#FE	allows an internal bit to be forced.
READ_GRAFCET_BIT	16#2A	16#5A	allows the status of 127 Graftet steps to be read.

### Access to objects of an I/O module

These requests provide a set of services allowing read/write access to the input/output data of the modules.

Name of request	Code request	Code report	Comment
READ_DIGITAL_MODULE_IMAGE	16#49	16#79	allows image bits for input/output of a single All or Nothing module to be read.
WRITE_DIGITAL_MODULE_IMAGE	16#4A	16#7A	allows image bits for input/output of a single All or Nothing module to be written.
READ_STATUS_MODULE	16#44	16#74	allows identification of the physical structure and complete status of a piece of equipment.
READ_IO_CHANNEL	16#43	16#73	allows input/output objects at channel level to be read (configuration data, status, ...).
WRITE_IO_CHANNEL	16#48	16#78	allows input/output objects at channel level to be written (configuration data, status, ...).

### Access to generic objects

These requests provide a set of read/write services for application, system, network management (error counter, ...), equipment management (configuration data, ...) objects.

Name of request	Code request	Code report	Comment
READ_OBJECT	16#36	16#66	allows one or more consecutive objects of the same type to be read.
WRITE_OBJECT	16#37	16#FE	allows one or more consecutive objects of the same type to be written.
WRITE_GENERIC_OBJECT	16#83	16#B3	allows all or part of a structured or simple object to be written.
READ_OBJECT_LIST	16#38	16#68	allows objects of different types to be read within the same request.
WRITE_OBJECT_LIST	16#39	16#69	allows objects of different types to be written within the same request.

Name of request	Code request	Code report	Comment
ACTION_GENERIC_OBJECT	16#9F	16#CF	allows management of operating modes, activation of specific communication operations, ...

### Management of operating modes

These requests provide a set of services which allow work on a processor's operating modes.

Name of request	Code request	Code report	Comment
RUN	16#24	16#FE	allows the processor's tasks to start operating.
STOP	16#25	16#FE	allows operation of the processor's tasks to be stopped.
INIT	16#33	16#63	allows activation of a cold or warm restart.

### Data transfer

These requests provide a set of services which allow the downloading or remote offloading of data between devices as well as the transfer or comparison of data within the same device.

Name of request	Code request	Code report	Comment
OPEN_DOWNLOAD	16#3A	16#6A	allows initialization of a loading phase of the data.
WRITE_DOWNLOAD	16#3B	16#6B	allows a data segment to be downloaded.
CLOSE_DOWNLOAD	16#3C	16#6C	allows a loading sequence to be terminated.
OPEN_UPLOAD	16#3D	16#6D	allows a backup phase to be initialized.
READ_UPLOAD	16#3E	16#6E	allows the contents of a file segment to be read during backup.
CLOSE_UPLOAD	16#3F	16#6F	allows a backup sequence to be terminated.
BACKUP	16#45	16#75	allows memory zones to be compared or memory zones to be backed up.

### Semaphore management

These requests provide a semaphore mechanism allowing protection of a server to avoid conflicting access during a critical period.

Name of request	Code request	Code report	Comment
RESERVE	16#1D	16#FE	allows a client to partially or globally keep the functions of a server.
RELEASE	16#1E	16#FE	allows client to release the reserved server.

---

Name of request	Code request	Code report	Comment
I_AM_ALIVE	16#2D	16#FE	necessary to maintain client's server reservation.

---

## Exchange of text data : DATA\_EXCH

---

### Introduction

The function `DATA_EXCH` allows data to be sent, data to be received or the linkage of both sending and receiving.

This function is used either to exchange data between two PL7 applications, or to send data to an item of equipment that has a specific application protocol.

---

### Syntax

The syntax of the communication function `DATA_EXCH` is presented in the following form:

```
DATA_EXCH(ADR#{20.2}APP, 2, %MW70:10, %MW80:1, %MW90:4)
```

The following table shows the different function parameters.

Parameter	Description
ADR#{20.2}APP	Address of the destination exchange item in sending mode or the address of the emitter in receipt mode. If it is an exchange operation, the broadcast addresses (ALL) are prohibited.
2	Type of operation: This parameter specifies the operation to be carried out: <ul style="list-style-type: none"><li>● 1: emission followed by a stand – by request for the receipt of a message ( unusable in the Uni-telway slave</li><li>● 2: single emission</li><li>● 3: stand – by request for the receipt of a message</li></ul>
%MW70:10	Data to send (word table containing the complete coding ). It must have a minimum length of one word even if there is not any data to be sent ( receipt operation ). The length of data to be sent( in bytes ) must be stored in the fourth word of the management parameter (word length) before starting this function.
%MW80:1	Table of words containing the value of data received. It must have a minimum length of one word even if there is not any data received (emission operation). The length of data actually received is indicated (in bytes), at the end of the exchange, in the fourth word of the management parameter.
%MW900:4	Management parameters. The operation report takes one of the following values: <ul style="list-style-type: none"><li>● 16#00: correct operation</li><li>● 16#02: incorrect operation</li><li>● 16#03: size of the incorrect response ( this value is insignificant if it is an emission operation)</li></ul>



**Note:** The fourth word in the management parameters table corresponds to the length parameter. It must contain the length (in bytes) of the data to be sent, before the function is executed. Once the request is executed it is automatically updated and contains the length of the response (in bytes).

### Programming rules

During the programming of a function `DATA_EXCH` the data that is to be sent must contain the complete coding of the frame that corresponds to the request.

Example of writing objects, carried out by the function `DATA_EXCH`. The data to be sent is coded in the following way:

Word	Byte 1 (most significant)	Byte 0 (least significant)	Comment
%MW70	16#07	16#37	<ul style="list-style-type: none"><li>● Byte 0: request code (writing object)</li><li>● Byte 1: category code</li></ul>
%MW71	16#07	16#68	<ul style="list-style-type: none"><li>● Byte 0: segment</li><li>● Byte 1: type of objects</li></ul>
%MW72	16#00	16#0A	Address of destination equipment
%MW73	16#00	16#0A	Quantity
%MW74 to %MW79	16#xx	16#xx	Value of data to be sent

## Exchange of text data : input Help screen

### Introduction

For this communication function, you can call up the input help screen.

### Function parameters

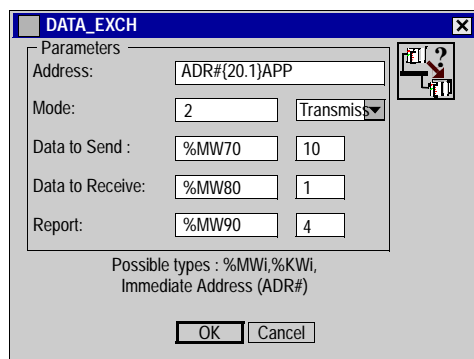
This function supports five parameters:

Parameters	Types of objects	Comments
Address	ADR# %MWX:n	If you input a value directly into the field, the Help button for the inputting of addresses is greyed out.
Mode	%MWx %KWx immediate value	In %MWx or %KWx, a value input field is displayed.
Data to be sent.	%MWx:n %KWx:n	-
Response	%MWx:n	-
Report	%MWx:4	-

**Note:** The symbols are accepted.

### Example

The following screen shows an example of inputting a function input:



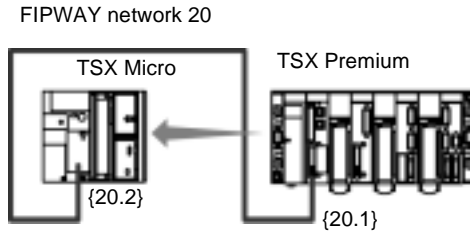
## Exchange of text data : examples of use

### Introduction

The example focuses on the sending of data to station 2 from network 20. The word %MW93 must contain the length of the data to be sent : %MW93 = 20 (10 words to send ).

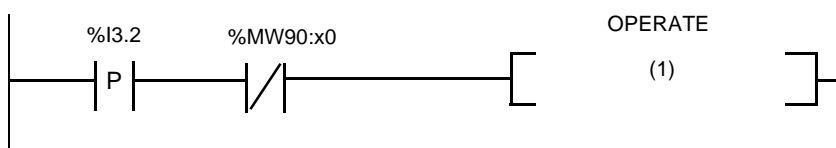
### Illustration

The two stations are connected via a FIPWAY network.



### Send

Programming of the function is as follows:



(1) DATA\_EXCH(ADR#{20.2}APP, 2, %MW70:10, %MW80:1, %MW90:4)

Query parameters:

Parameters	Description
ADR#{20.2}APP	<ul style="list-style-type: none"> <li>{20.2}: network 20, station 2</li> <li>APP: application</li> </ul>
2	Request to send
%MW70:10	Data sent (for the example: 10 words to send )
%MW80:1	Contents of the response ( for the example: no data to receive)
%MW90:4	Report

**Note:** Before starting the function each time, initialize the length parameter (in the example: %MW93 = 20 octets).

Receipt

Station 2 of network 20 receives data sent by station 1. Programming of the function is as follows:



(1) DATA\_EXCH(ADR#{20.2}APP, 3, %MW70:10, %MW80:1, %MW90:4)

Query parameters:

Parameters	Description
ADR#{20.2}APP	<ul style="list-style-type: none"><li>● {20.2}: network 20, station 2</li><li>● APP: application</li></ul>
3	Receipt request
%MW70:1	Contents of the data to be sent (for the example: no data to send)
%MW80:1	Data received (for the example: 10 words to receive )
%MW90:4	Report

**Note:** Before starting the function each time, initialize the length parameter (in the example: %MW93 = 0 bytes).

## Text Type Data Exchange: examples of use with an Altivar

### At a Glance

This example deals with a Uni-Telway transmission request (request code 16#F2) by a TSX Premium PLC to an Altivar ATV58 (address Ad0 = 6).

The `SEND_REQ` communication function with request code 16#F2 does not work, the PLC cannot read the report.

The solution is to send the `DATA_EXCH` function in send/receive.

### Send

Syntax as follows:

```
DATA_EXCH(ADR#0.1.6, 1, %MW100:6, %MW200:6, %MW250:4)
```

Parameters of the request:

Parameters	Description
ADR#0.1.6	<ul style="list-style-type: none"> <li>0 : module</li> <li>1 : channel 1</li> <li>6 : Destination address Ad0</li> </ul>
1	Operation type: transmission followed by "await reception" request
%MW100:6	Table to send
%MW200:6	Table to receive
%MW250:4	Report

**Note:** Before each function launch, initialize the length parameter (%MW253 = 12 octets in the example).

### Table Structure

The following table shows the byte table to be sent:

Parameters	Description
%MW100	<ul style="list-style-type: none"> <li>Byte 0: request code 16#F2</li> <li>Byte 1: category code 16#07</li> </ul>
%MW101	16#0000 Value
%MW102	Command word <code>CMD</code>
%MW103	Setpoint word <code>FRH</code>
%MW104	Acceleration adjustment <code>ACC</code>
%MW105	Deceleration adjustments <code>DEC</code>

The following table shows the byte table to be received:

Parameters	Description
%MW200	Response Code 16#37F2
%MW201	<ul style="list-style-type: none"><li>● Byte 0: Value 16#00</li><li>● Byte 1: most significant setpoint byte FRH</li></ul>
%MW202	<ul style="list-style-type: none"><li>● Byte 0: least significant setpoint byte FRH</li><li>● Byte 1: most significant status register byte ETA</li></ul>
%MW203	<ul style="list-style-type: none"><li>● Byte 0: least significant status register byte ETA</li><li>● Byte 1: most significant fault register byte FLT</li></ul>
%MW204	<ul style="list-style-type: none"><li>● Byte 0: least significant fault register byte FLT</li><li>● Byte 1: most significant motor current byte LCR</li></ul>
%MW205	<ul style="list-style-type: none"><li>● Byte 0: least significant motor current byte LCR</li><li>● Byte 1: Value 16#00</li></ul>

---

## Sending a telegram : SEND\_TLG

### Introduction

Function `SEND_TLG` allows telegram data to be sent to a remote PL7 application.

Data to be sent has a maximum length of 16 bytes. Unlike the other communication functions, it is processed immediately (synchronous). The activity bit parameters and timeout therefore do not exist.

Consequently the word table assigned to the management parameters only uses two words instead of four. This function is only usable in the processor on FIPWAY, and for the address stations 0 to 15.

### Syntax

The syntax of the communication function `SEND_TLG` is presented in the following format:

```
SEND_TLG(ADR#{20.3}APP, %MW190:8, %MW200:2)
```

The following table shows the different function parameters.

Parameter	Description
ADR#{20.3}APP	Address of the exchange destination item Only the addresses{network.Station}APP.number are authorized.
%MW190:8	Data to be sent. This word table has a maximum length of 8 words ( 16 bytes).
%MW200:2	Management parameters. The word table must contain : the communication report and the length of the data to send. The operation report takes one of the following values: <ul style="list-style-type: none"> <li>● 16#00: correct exchange</li> <li>● 16#03: Incorrect address format</li> <li>● 16#04: Incorrect target address</li> <li>● 16#05: Incorrect management parameters</li> <li>● 16#06: Incorrect specific parameters</li> <li>● 16#07: Module at fault</li> <li>● 16#0A: Size of emission buffer is insufficient</li> <li>● 16#0B: Absence of resource system</li> </ul>

**Note:** The length of data sent must be stored( in bytes) in the second word of the management parameter (word length) before starting this function.

## Sending a telegram: entry help screen

**At a Glance** For this communication function, you can request help from the entry help screen.

**Function parameters** This function supports three parameters:

Parameters:	Type of Object	Comments
Address	ADR# %MWX:n	If you enter a value directly into the field, the address entry help button grays out.
Data to send	%MWx:n %KWx:n	The number of words lies between 1 and 8.
Report	%MWx:2	The number of words for the management parameters is 2.

**Note:** Symbols are accepted.

**Example** The following screen shows an example of function entry:

**SEND\_TLG**

Parameters

Address: ADR#{20.3}APP

Data to send: %MW190 8

Report: %MW200 2

Accepted types: %MWi:n (n=2)

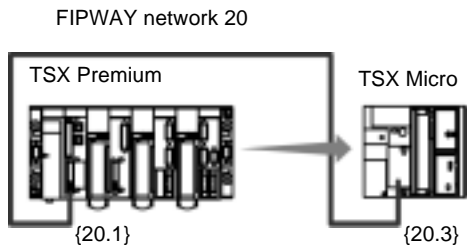
OK Cancel



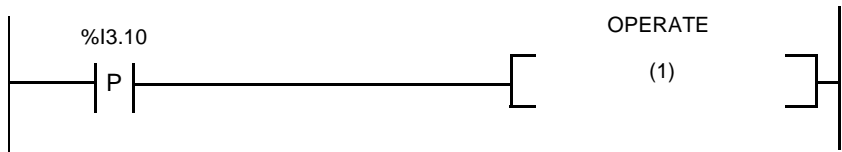
## Sending a telegram : example of use

**Introduction** The example shows a telegram with 8 words being sent from station 1 to remote station 3 on the FIPWAY 20 network.

**Illustration** The two stations are connected via a FIPWAY network.



**Send** Programming of the function is as follows:



(1) SEND\_TLG(ADR#{20.3}APP, %MW180:8, %MW200:2)

Query parameters:

Parameters	Description
ADR#{20.3}APP	<ul style="list-style-type: none"><li>• {20.2}: network 20, station 3</li><li>• APP: application</li></ul>
%MW190:8	Contents of the telegram to be sent
%MW200:2	Report

**Note:** The word %MW 201 must be initialized to 16 (8 words) before sending the request.  
Synchronous implementation of this function requires the immediate testing of the operation report after the program line that activates the implementation of this function.

## Receiving a telegram : RCV\_TLG

---

### Introduction

Function `RCV_TLG` allows the telegram data coming from a remote PL7 application to be read.

Data to be received has a maximum length of 16 bytes. Unlike the other communication functions, it is processed immediately (synchronous). The activity bit parameters and timeout therefore do not exist.

Consequently the word table assigned to the management parameters only uses two words instead of four. It can be activated in a factual task, in the FAST task or the MAST task. This function is only usable in the processor on FIPWAY, and for the address stations 0 to15.

---

### Syntax

The syntax of the communication function `RCV_TLG` is presented in the following format:

```
RCV_TLG(%MW300, %MW310:8, %MW320:2)
```

The following table shows the different function parameters.

Parameter	Description
%MW300	Address of the station sending the telegram when the message has been received. The least significant byte corresponds to the number of the network in hexadecimal. The most significant byte corresponds to the number of the station in hexadecimal.
%MW310:8	Reception buffer. Word table containing the data received. The maximum length of this table is 8 words ( 16 bytes).
%MW320:2	Management parameters. The table with two words must contain : <ul style="list-style-type: none"><li>● for the first word, the communication and operation report,</li><li>● for the second word, the length of the data actually received.</li></ul> The operation report takes one of the following values: <ul style="list-style-type: none"><li>● 16#00: correct exchange</li><li>● 16#05: incorrect management parameters</li><li>● 16#06: incorrect specific parameters</li><li>● 16#09: size of reception buffer is insufficient</li><li>● 16#0B: absence of resource system</li><li>● 16#0D: no telegram received</li><li>● 16#10: network coupler absent</li><li>● 16#0F: telegram service non-configured</li></ul>

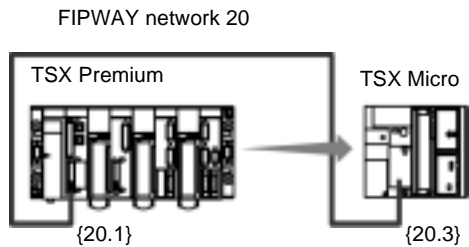
<b>Note:</b> Length of data received is indicated in bytes (maximum 16).
--

---

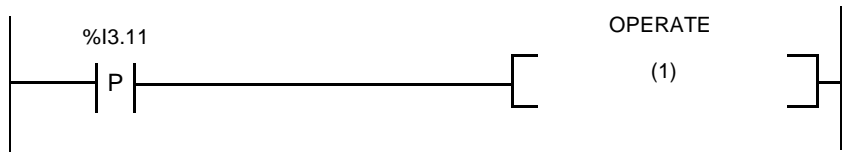
## Receiving a telegram : example of use

**Introduction** The example shows the receipt of a telegram with 8 words (16 bytes) from a remote application.

**Illustration** The two stations are connected via a FIPWAY network.



**Receive** Programming of the function is as follows :



(1) RCV\_TLG(%MW300, %MW310:8, %MW320:2)

Query parameters :

Parameters	Description
%MW300	Contains the address of the sender at the end of the exchange.
%MW310:8	Contents of the telegram received
%MW200:2	Report

**Note:** When a function RCV\_TLG is programmed in a factual task, it can not be used in the MAST or FAST task.  
Synchronous implementation of this function requires the immediate testing of the operation report after the program line that activates the implementation of this function.

## Writing a string of characters. PRINT\_CHAR

### Introduction

Function `PRINT_CHAR` allows the sending of a chain of characters of 4 K bytes, 120 bytes on the terminal port, destined to be sent on a character mode link or to the PLC terminal port.

A message can also be transmitted in immediate value form (series of bytes between apostrophes, example : 'Message to send').

Special characters can also be transmitted, they must then start with the \$ character followed by the hexadecimal value of the character to be sent, example \$0D.

Another possibility is that the special characters are defined as : \$R= CR ( return carriage), \$L = LF (return to the line ), \$N = CR+LF.

### Syntax

The syntax of the communication function `DATA_EXCH` is presented in the following format :

```
PRINT_CHAR(ADR#{20.3}2.0.SYS, 'Overheating oven 4$L$R',
%MW110:4)
```

The following table shows the different function parameters.

Parameter	Description
ADR#{20.3}2.0.SYS	Address of the character mode channel that is sending the message. Only the system addresses (which finish with SYS) are supported by this function ( example : { Network.Station}Rack module.Channel.SYS).
'Oven overheating 4\$L\$R'	Character string to be sent. It is either stored in a byte table (%MB..) or provided with an immediate value. Its length is at a maximum : <ul style="list-style-type: none"> <li>• 4 Kbytes to a character chain link managed by a PCMCIA TSX SCP11 card • and via the built-in link in module TSX SCY 2160 •,</li> <li>• 120 bytes to a terminal point,</li> <li>• 250 bytes if the value is provided in immediate value.</li> </ul>
%MW900:4	Management parameters. The operation report takes one of the following values : <ul style="list-style-type: none"> <li>• 16#00 : correct operation</li> <li>• 16#01 : operating error</li> <li>• 16#02 : incorrect operation</li> <li>• 16#04 : signals error RTS/CTS</li> </ul>

**Note:** The length of data sent is stored in the fourth word of the management parameter (word length) before starting this function. If the length is initialized at 0, the whole of the character string is sent.  
After sending the characters, it contains the number of characters sent in the form of a communication report.

## Programming rules

The special characters are preceded by the character \$ in the string to be sent. Characters \$ are not sent by the sender, they must therefore not be counted during the initialization of the length parameter.

The spaces between two characters are the equivalent of a byte.

So in the example `Oven overheating 4$L$R`, the length of the data to send is 19 bytes.

When sending a character string longer than 240 bytes, it will need several PLC cycles ( there is fragmentation of the string ).

It is, therefore, important that you ensure that the management data is not modified during function processing. The system guarantees that the string will be sent coherently in several fragments, but it does not prohibit the sending of another character string between two fragments.

On a terminal port link configured in character mode , if the function `PRINT_CHAR` is activated whilst the function `INPUT_CHAR` is running, the function `PRINT_CHAR` stays blocked. It is recommended that you program a timeout to occur in these functions.

## Writing a chain of characters : input help screen

### Introduction

For this communication function, you can call up the input help screen.

### Function parameters

This function supports six parameters:

Parameters	Types of objects	Comments
Address	ADR# %MWX:n	If you input a value directly into the field, the Help button for the inputting of addresses is greyed out.
Chain to send	%MBx:n %KBx:n Immediate value	In %KBx:n, a value input field is displayed.
Report	%MWx:2	There are two words for the management parameter.

**Note:** The symbols are accepted.

### Example

The following screen shows an example of input function:

**PRINT\_CHAR**

Parameters

Address : ADR#{20.3}2.0.SYS

String to Send

Variable :

Value : oven overheat 4\$L\$R

Report : %MW110 4

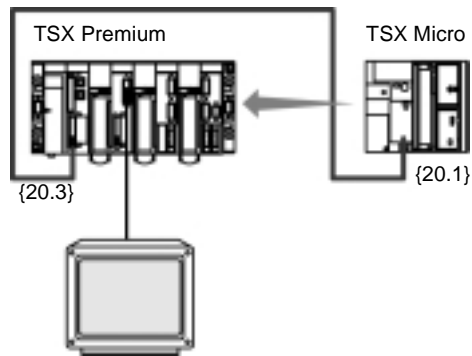
Possible types : %MWi:n (n=4)

OK Cancel

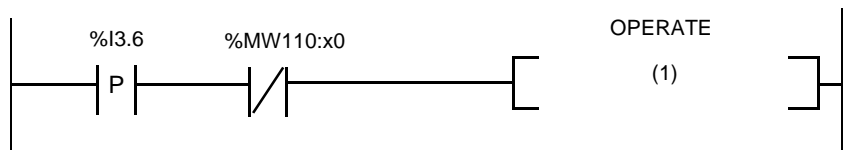
## Writing of a chain of characters : example of use

**Introduction** The example shows the sending of a chain of characters to a video terminal connected via a link integrated from the module TSX SCY 21600 / 21601 of a PLC with the address network 20, station 3.

**Illustration** The two stations are connected via a FIPWAY network.  
FIPWAY network 20



**Send** Programming of the function is as follows :



(1) PRINT\_CHAR(ADR#{20.3}2.0.SYS, 'Overheating oven 4\$L\$R', %MW110:4)

Query parameters

Parameters	Description
ADR#{20.3}2.0.SYS	<ul style="list-style-type: none"><li>● {20.3} : network 20, station 3</li><li>● 2 : module</li><li>● 0 : channel 0</li><li>● SYS :system address</li></ul>
'Oven overheating 4\$L\$R'	Chain of characters to be sent ( input in immediate value)
%MW50:4	Report



**Note:** Before starting the function each time, initialize the length parameter (in the example: %MW113 = 19 ).

---

## Reading a character string: INPUT\_CHAR

---

### At a Glance

The function `INPUT_CHAR` enables a request to be sent to a character mode communication module to read a character string. The received message is saved in a byte table `%MB`.

Up to 4 KB can be received with this function (120 bytes for the terminal block). It should be consistent with the channel configuration, otherwise an error will reoccur.

Two exclusive possibilities are offered:

- reading a number of characters: no condition must be configured,
- reading a message: a stop condition must be configured in the configuration screen.

### Syntax

The syntax for the communication function `INPUT_CHAR` is as follows:

```
INPUT_CHAR(ADR#{20.5}0.0.SYS, 1, 0, %MB200:20, %MW120:4)
```

The following table describes the different function parameters.

Parameter	Description
ADR#{20.5}0.0.SYS	Address of the character mode channel receiving the message. Only the system addresses (which end in SYS) are supported by this function (example: {Network.Station}RackModule.Channel.SYS).
1	<p>Reset to 0. This parameter specifies that the coupler receiver memory is reset to zero.</p> <ul style="list-style-type: none"><li>● value at 0: the memory is not reset to zero,</li><li>● value at 1: memory is reset to zero.</li></ul> <p><b>Note:</b> For communication with the terminal block, the value has to be 1.</p>
0	<p>Number of characters:</p> <ul style="list-style-type: none"><li>● value at 0: reading an available message, in this case a stop condition must be specified in the configuration screen,</li><li>● value above 0: defines the number of characters to be read.</li></ul> <p><b>Note:</b> For communication with the terminal block, only the value 0 is permitted. In this case, the message end character is:</p> <ul style="list-style-type: none"><li>● by default, a carriage return (CR) for a Premium PLC,</li><li>● the character configured on the screen for a Micro PLC.</li></ul>
%MB200:20	String received. It is saved in a byte table (%MB..).

Parameter	Description
%MW120:4	<p>Management parameters. The length of the received data is saved in the last at the end of the execution of this function.</p> <p>The operation report takes one of the following values:</p> <ul style="list-style-type: none"><li>● 16#00 : correct operation</li><li>● 16#01 : operation error</li><li>● 16#02 : incorrect operation</li><li>● 16#03 : size of the incorrect response</li><li>● 16#06 : coupler configured in character mode</li><li>● 16#07 : coupler configured in message mode</li></ul>

### Programming rules

When several `INPUT_CHAR` are launched simultaneously, the Reset to 0 parameter must be positioned (the receiving coupler memory is not reset to zero). Resetting the coupler memory to zero can be requested for the next message in order to avoid receiving old data.

When Reset is at 1, `INPUT_CHAR` must be launched before sending the data.

Receiving a character string of more than 240 bytes requires several PLC cycles (there is string fragmentation). Therefore, it is important to ensure that the management data are not modified during the function processing. The system guarantees the coherent reception of the string in several fragments.

## Reading a string of characters : input Help screen

**Introduction** For this communication function, you can call up the input Help screen.

**Function parameters** This function supports three parameters:

Parameters	Types of objects	Comments
Address	ADR# %MWx:n	If you input a value directly into the field, the Help button for the inputting of addresses is greyed out.
reset module memory	%MWx %KWx Immediate value	The selection of <i>Yes/No</i> directly displays the immediate value 1 / 0.
Number of characters	%MWx %KWx Immediate value	When this number equals 0, the whole following message will be received.
String to be sent	%MBx:n	-
Report	%MWx:4	-

**Note:** The symbols are accepted.

**Example** The following screen shows an example of the function input:

**INPUT\_CHAR**

Parameters

Address : ADR#{21.5}0.0.SYS

Reset Module Memory : 1 ☒ Yes ☐ No

Type of Read

☒ Read Message with Stop Condition  
☐ Define Number of Characters to Read

Number of Characters : 0

String to Receive : %MB200 20

Report : %MW120 4

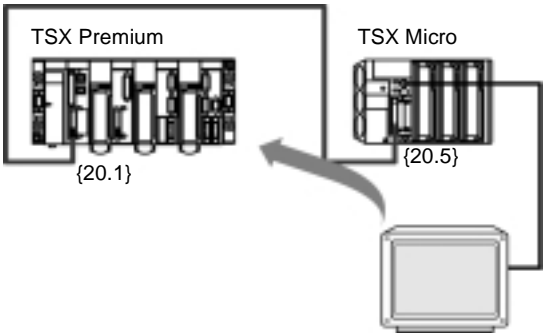
Possible types : %MBi:n (i must be even)

OK Cancel

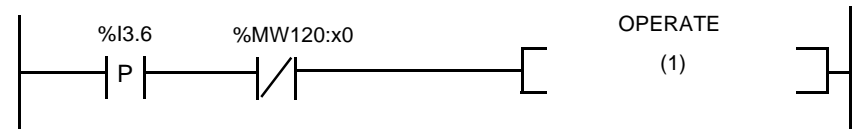
## Reading a character string : example of use

**Introduction** The example deals with the reading of a string of characters sent by a video terminal plugged into the TER port in the PLC with network address 20, station 5.

**Illustration** The two stations are connected via a FIPWAY network.  
FIPWAY network 20



**Receive** Programming of the function is as follows :



(1) INPUT\_CHAR(ADR#{20.5}0.0.SYS, 1, 0, %MB200:20, %MW120:4)

Query parameters :

Parameters	Description
ADR#{20.5}0.0.SYS	<ul style="list-style-type: none"><li>● {20.5} : network 20, station 5</li><li>● 0 : module</li><li>● 0 : channel 0</li><li>● SYS : system address</li></ul>
1	Reset to zero
0	Reading of the whole string of characters
%MB200:20	Contents of the message in bytes
%MW110:4	Report

**Note:** Each time before each starting of the function, initialize the length parameter (in the example : %MW113 = 19).

## **Sending/receiving a string of characters : OUT\_IN\_CHAR**

---

### **Introduction**

The function `OUT_IN_CHAR` allows the sending of a string of a maximum of 240 bytes (120 bytes for the terminal port) followed by a receipt message call (the sending alone or the reception alone are equally possible).

A message can also be transmitted in immediate value form (series of words between apostrophes, example : 'Message to send').

Special characters can also be transmitted, they must then start with the \$ character followed by the hexadecimal value of the character to be sent, example \$0D.

Another possibility is that the special characters are defined as : \$R= CR (carriage return), \$L = LF (new line), \$N = CR+LF.

Although the principal use of this function is the communication with a terminal port, it can be used in another link of the character mode type. On receiving a message request, the destination coupler resets its receive memory. It is mandatory to configure an end message or pause in the configuration screen.

---

**Syntax**

The syntax of the communication function `OUT_IN_CHAR` is presented in the following format :

```
OUT_IN_CHAR(ADR#{20.5}0.0.SYS, 1, %MB300:10, %MB310:10,
%MW170:4)
```

The following table describes the different parameters of the function.

Parameter	Description
ADR#{20.5}2.0 .SYS	Address of the character mode channel that is sending the message. Only the system addresses (which finish with SYS) are supported by this function (example : {Network.Station}RackModule.Channel.SYS).
1	Mode. This parameter specifies the operation mode : <ul style="list-style-type: none"> <li>● 1 : sending a message and receive request</li> <li>● 2 : sending a message</li> <li>● 3 : receive message request</li> </ul>
%MB300:10	Character string to be sent. This table must have a minimum length of one character, even if there is no data to be sent.
%MB310:10	Character string or message received. This table must have a minimum length of one character, even if there is no data to receive.
%MW170:4	Management parameters. The operation report takes one of the following values: <ul style="list-style-type: none"> <li>● 16#00 : correct operation</li> <li>● 16#01 : operating error</li> <li>● 16#02 : incorrect operation</li> <li>● 16#03 : size of the incorrect response</li> <li>● 16#04 : signals error RTS/CTS</li> <li>● 16#06 : coupler configured without stop condition</li> <li>● 16#08 : draft error</li> </ul>

**Note:** The length of data sent is stored in the fourth word of the management parameter (word length) before starting this function. If the length is initialized at 0, the whole of the character string is sent.  
The length of the received data is stored in the fourth word of the management parameter (length word). A stop condition must have been configured.

## Sending/receiving a string of characters : input Help screen

**Introduction** For this communication function, you can call up the input Help screen.

**Function parameters** This function supports three parameters :

Parameters	Types of objects	Comments
Address	ADR# %MWx:n	If you input a value directly into the field, the Help button for the inputting of addresses is greyed out.
Mode	%MWx %KWx Immediate value	Selecting from the drop-down list Exchange, Send, Receive directly displays the immediate value 1, 2 or 3.
String to be sent	%MBx:n %KBx:n Immediate value	If the input of the string is carried out by %MBx:n, the field of the immediate value disappears.
String to receive	%MBx:n	-
Report	%MWx:4	-

**Note:** The symbols are accepted.

**Example** The following screen shows an example of function input:

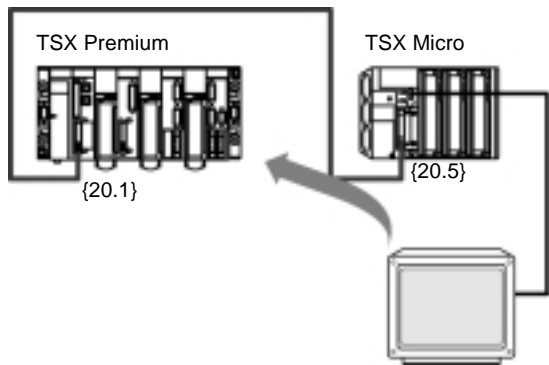
The screenshot shows a dialog box titled "OUT\_IN\_CHAR" with a close button (X) in the top right corner. Inside the dialog, there is a "Parameters" section. Under "Address", the text "ADR#[20.5]0.0.SYS" is entered in a text field. To the right of this field is a "HELP ?" button with a question mark icon. Below "Address" is the "Mode" section, which has a text field containing "1" and a dropdown menu currently showing "Exchange". Below "Mode" is the "String to Send" section, which contains a "Variable" label, a text field with "%MB300", and a text field with "10". Below "String to Send" is the "String to Receive" section, which has a text field with "%MB310" and a text field with "10". Below "String to Receive" is the "Report" section, which has a text field with "%MW170" and a text field with "4". At the bottom of the dialog, there is a label "Possible types : %MWi:n (n=4)" and two buttons: "OK" and "Cancel".



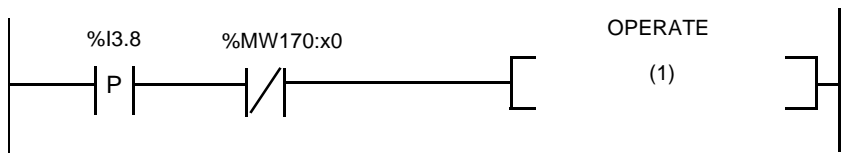
## Sending/receiving a string of characters : example of use

**Introduction** The example deals with the sending/receiving of a character string to and from a video terminal connected to the TER port configured in character mode.

**Illustration** The two stations are connected via a FIPWAY network.  
FIPWAY network 20



**Programming** Programming of the function is as follows :



(1) OUT\_IN\_CHAR(ADR#{20.5}0.0.SYS, 1, %MB300:10, %MB310:10, %MW170:4)

Query parameters :

Parameters	Description
ADR#{20.5}0.0.SYS	<ul style="list-style-type: none"><li>• {20.5} : network 20, station 5</li><li>• 0 : module</li><li>• 0 : channel 0</li><li>• SYS : system address</li></ul>
1	Send / receive
%MB300:10	Contents of the message to be sent in bytes
%MB310:10	Contents of the message received in bytes
%MW170:4	Report

**Note:** Each time before each starting of the function, initialize the length parameter (in the example : %MW173 = 10).  
After the exchange, it will contain the length of the received data.

---

---

## Stopping an exchange whilst in action. : CANCEL

---

### Introduction

The function `CANCEL` allows the interruption of an asynchronous function whilst in action. The exchange number assigned at the beginning of the transaction allows identification of the function to be halted.

The execution of this function is synchronous with the execution of program PL7.

---

### Syntax

The syntax of the communication function `CANCEL` is presented in the following format :

```
CANCEL(%MW180, %MW185)
```

The following table describes the different parameters of the function.

Parameter	Description
%MW180	Exchange number. This parameter specifies the exchange number whose execution must be interrupted. <b>Note :</b> The exchange number is indicated in the most significant byte of the first word in the management parameters of the function to be interrupted.
%MW185	Management parameters. The operation report takes one of the following values : <ul style="list-style-type: none"><li>● 16#00 : correct operation. Communication is interrupted, the activity bit of the interrupted function is set to 0 and its report takes the value 2.</li><li>● 16#0C : incorrect exchange number</li></ul>

---

## Stopping an exchange whilst in action : example of use

### Introduction

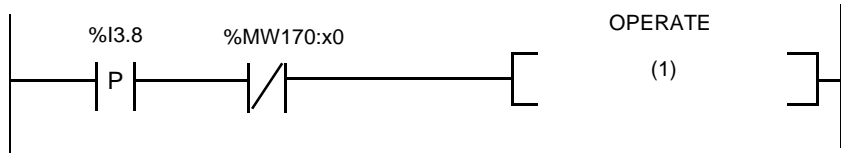
The example deals with the stopping of the communication function `OUT_IN_CAR`.

When starting the function `OUT_IN_CHAR`, an exchange number is automatically assigned to it. This number distinguishes the exchange until the end of the operation. The function `CANCEL` uses this number to interrupt this operation.

### Sending the function

`OUT_IN_CHAR`

Programming of the function is as follows :



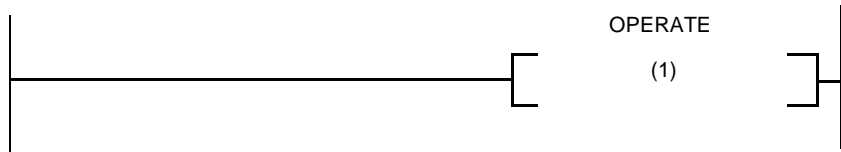
(1) `OUT_IN_CHAR(ADR#{20.5}0.0.SYS, 1, %MB300:10, %MB310:10, %MW170:4)`

### Sending the function

`CANCEL`

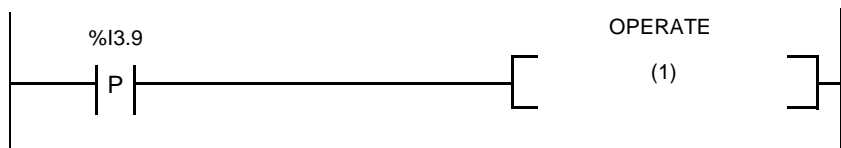
To prepare the cancellation of the exchange `OUT_IN_CHAR` it is necessary to make a shift of 8 bits to put the most significant bit of `%MW170` in `%MW180` for example. `%MW180` will be used by the function `CANCEL`, and will contain the exchange number of the function `OUT_IN_CHAR`.

Programming the shift of the 8 bits is as follows :



(1) `%MW180:=SHR(%MW170,8)`

Programming of the function is as follows :



(1) `CANCEL(%MW180, %MW185)`

Query parameters :

Parameters	Description
%MW180	Contains the exchange number of the function <code>OU_IN_CHAR</code> to be interrupted.
%MW185	Report

---

## Shifting a byte to the right in a table : ROR1\_ARB

---

### Introduction

The function `ROR1_ARB` allows a byte to be circularly shifted to the right in a table of bytes. It is used after receiving a response to certain UNI-TE requests (function `SEND_REQ` for example).

The function `ROR1_ARB` does not carry out communication processing, on the other hand, it is necessary to process certain UNI-TE requests (for example, reading a word table with the function `SEND_REQ`).

### Syntax

The syntax of the communication function `ROR1_ARB` is presented in the following format :

```
ROR1_ARB ( %MB420 : 12 )
```

The following table describes the different parameters of the function.

Parameter	Description
%MB420 : 12	This byte table specifies the address of the first word in the table and the number of bytes contained in the table to be shifted.

---

## Shifting a byte to the right in a table : example of use

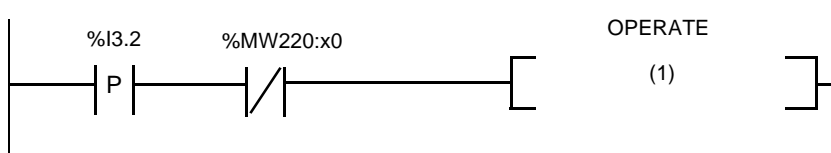
### Introduction

The example deals with the shifting of a table following the reading of a table of 5 words in the PLC with network address 2 and station address 4.

The function of reading objects is carried out by the communication function `SEND_REQ` and the request code = 36 (16#0036).

### Sending the function `SEND_REQ`

Programming of the function is as follows :



(1) `SEND_REQ(ADR#{2.4}SYS, 16#0036, %MW200:6, %MW210:6, %MW220:4)`

The send table is as follows :

Word	Value	Description
%MW200	16#0768	<ul style="list-style-type: none"> <li>16#07 : type of object (16 bit integer).</li> <li>16#68 : segment (internal words).</li> </ul>
%MW201	50	Source of the word table to be read (decimal value).
%MW202	05	Number of words to be read (decimal value).

The table of words read is as follows :

Word	Byte 1	Byte 0
%MW210 :=	Least significant bit of first word read	16#07 (type of object)
%MW211 :=	Least significant bit of second word read	Most significant bit of first word read
%MW212 :=	Least significant bit of third word read	Most significant bit of second word read
%MW213 :=	Least significant bit of fourth word read	Most significant bit of third word read
%MW214 :=	Least significant bit of fifth word read	Most significant bit of forth word read
%MW215 :=	Not significant	Most significant bit of fifth word read

The least significant byte of the first word read contains the type of objects read, the reception table is thus shifted by 1 byte.

---

**use of the  
ROR1\_ARB  
function**

To re-categorize the receive table, it is necessary to make a one byte right shift by means of the function ROR1\_ARB.

The receive table becomes :

Word	Byte 1	Byte 0
%MW210 :=	Most significant bit of first word read	Least significant bit of first word read
%MW211 :=	Most significant bit of second word read	Least significant bit of second word read
%MW212 :=	Most significant bit of third word read	Least significant bit of third word read
%MW213 :=	Most significant bit of fourth word read	Least significant bit of fourth word read
%MW214 :=	Most significant bit of fifth word read	Least significant bit of fifth word read
%MW215 :=	Not significant	16#07 (type of object)

---



---

## Swapping bytes in a word table : SWAP

---

### Introduction

The function *SWAP* allows, in a word table, the inverting of all the least significant bytes and the most significant bytes.

This facilitates the management of messages, especially using Modbus.

---

### Syntax

The syntax of the communication function *SWAP* is presented in the following format :

*SWAP* ( %MW40 : 4 )

The following table describes the different parameters of the function.

Parameter	Description
%MW40 :	This byte table is defined by : <ul style="list-style-type: none"><li>● %MW40 : first word in table to be inverted</li><li>● 4 : number of words to be inverted</li></ul>

---

### Example

The table of words to be inverted is as follows :

Word	Value
%MW40 :=	16#0F43
%MW41 :=	16#21AC
%MW42 :=	16#8127
%MW43 :=	16#8811

After the application of the function *SWAP* , the table is as follows :

Word	Value
%MW40 :=	16#430F
%MW41 :=	16#AC21
%MW42 :=	16#2781
%MW43 :=	16#1188

---

## Reading of common Modbus Plus data: READ\_GDATA

---

### Introduction

A TSX Micro or Premium PLC can use global data exchange to communicate with remote Modbus stations.

Global data is a data base shared between a maximum of 64 stations on a network . Each station can write 32 words that are usable by all the other stations, and can read 32 words from each remote station.

Function `READ_GDATA` allows the reading of 32n words from a remote station.

---

### Syntax

The syntax of the communication function `READ_GDATA` is presented in the following form :

```
READ_GDATA(ADR#0.1.10, %MW100:32, %MW200:4)
```

The following table describes the different parameters of the function.

Parameter	Description
ADR#0.1.10	Address of the exchange destination item
%MW100:32	Address of the receipt zone of the global data
%MW200:4	Report

**Note:** The address of the target item is initialized with the node value to which the station containing the objects to be read is linked.

It is not necessary to initialize the length parameter before starting the function. At the end of the operation, the management word contains the size in number of bytes of the global data produced by the station specified in the address.

---

## Writing of Modbus Plus common data : WRITE\_GDATA

### Presentation

A TSX Micro or Premium PLC can use global data exchange to communicate with remote Modbus stations.

Global data is a data base shared between a maximum of 64 stations on a network . Each station can write 32 words that are usable by all the other stations, and can read 32 words from each remote station.

Function `WRITE_GDATA` allows 32 words from a remote station to be written.

### Syntax

The syntax of the communication function `WRITE_GDATA` is presented in the following form :

`WRITE_GDATA(ADR#0.1.SYS, %MW100:32, %MW200:4)`

The following table describes the different parameters of the function.

Parameter	Description
ADR#0.1.SYS	System address of the local PCMCIA card <b>Note :</b> The parameter of the receipt address must be initialized with the value of the address of the local Modbus Plus server.
%MW100:32	Address of the zone containing the words to be produced
%MW200:4	Report

**Note:** The function is defined to transfer 32 internal words %MW from the PL7 application to the common data buffer in the PCMCIA card. A buffer of a maximum of 32 words contains the data. All the contents of the buffer will be recopied into the global data base. The word indicating the length is not used.

## Immediate server : **SERVER**

---

**Presentation**      Function `SERVER` allows the `UNI_TE` requests to be processed immediately from the application program.

This function can be activated in the `MAST` task or in the `FAST` task.

At any given moment only one function `SERVER` can be activated by the application.

---

**Availability**      It can only be used to process the requests from a Modbus link (PCMCIA TSX SCP 114 card in the TSX SCY 21601 module, configured in Modbus slave with an immediate server).

---

**Syntax**            The syntax of the communication function `SERVER` is presented in the following format :

```
SERVER(ADR#{20.3}APP, %MW190, %MW200:2)
```

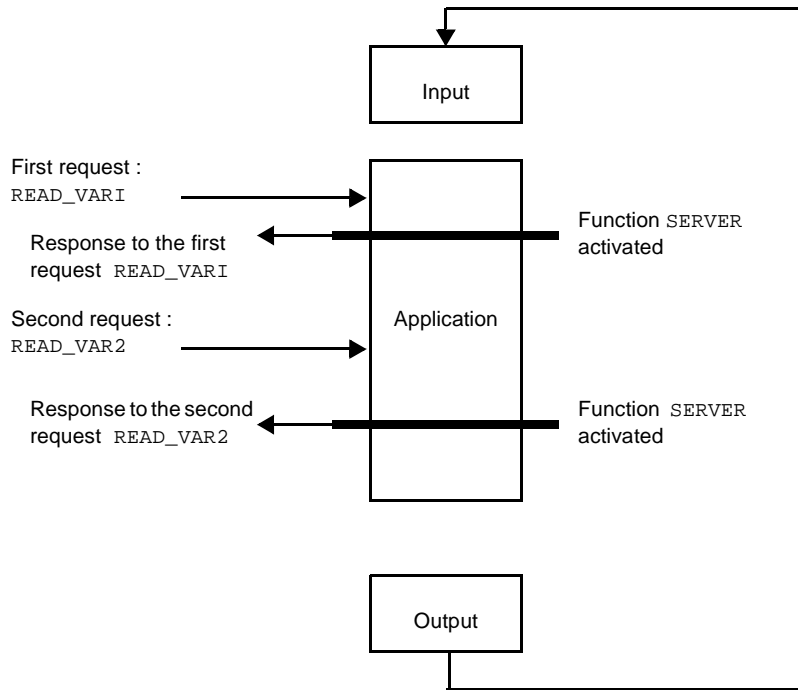
The following table describes the different parameters of the function.

Parameter	Description
ADR#{20.3}APP	Address of the exchange destination. Table of three words to store the address {Network.Station}Port.Module.Module channel to which the request transmitter is linked.
%MW190	Request response. This word allows the storage of the received request code (least significant byte), and the returned response code (least significant byte).
%MW200:2	Management parameters. The table with two words must contain : <ul style="list-style-type: none"><li>• for the first word, the exchange number and activity bit,</li><li>• for the second word, the communication and operation report.</li></ul> The operation report takes one of the following values : <ul style="list-style-type: none"><li>• 16#00 : Correct exchange</li><li>• 16#01 : Timeout, the response has not been able to be sent in less than 2 seconds</li><li>• 16#02 : Stop on user request (STOP,S0,INIT,a cold or warm restart)</li><li>• 16#03 : Incorrect address format</li><li>• 16#05 : Incorrect management parameters</li><li>• 16#07 : Problem with sending to destination</li><li>• 16#11 : No request received</li><li>• 16#12 : Function <code>SERVER</code> already called by another task</li><li>• 16#FF : Message refused</li></ul>

**Note:** When the communication report has the value 16#FF (message refused) an error has been detected. The operation report (most significant byte) can therefore take the value 16#14 ( server stopped).

## Exchange principal

the following diagram shows exchanges when using the communication function SERVER .



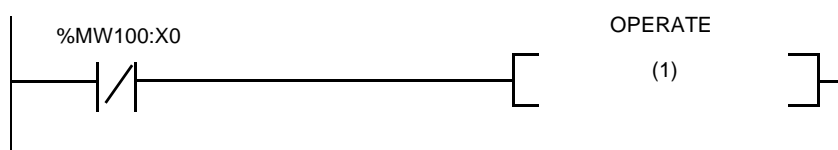
## Immediate server : example of use

## Introduction

The communication function `SERVER` responds to a request `READ_VAR` about the reading of x amount of words (corresponds to the code 16#03 according to the Modbus protocol ).

**Send**

Programming of the function is as follows :



(1) SERVER (%MW0:3, %MW10, %MW100:2)

Query parameters :

Parameters	Description
%MW0:3	This table contains the address of the request transmitter (network, station, port, module, channel ).
%MW10	Request response. <ul style="list-style-type: none"> <li>● Byte 1 = 16#03 : request code for the reading of x amount of words by Modbus</li> <li>● Byte 0 = 16#03 : Response code to the received request for the reading of x amount of words by Modbus</li> </ul>
%MW100:2	Report

## Asynchronous messaging service : **WRITE\_Asyn** and **READ\_Asyn**

---

### Introduction

These two functions allow the TSX ETY 110 coupler's asynchronous electronic messaging channel to write or to read 1 Kbytes of PL7 objects.

Functions `WRITE_Asyn` and `READ_Asyn` are only sent at the end of the MAST task , if this task is configured in periodic mode. It is possible to simultaneously activate 8 functions.

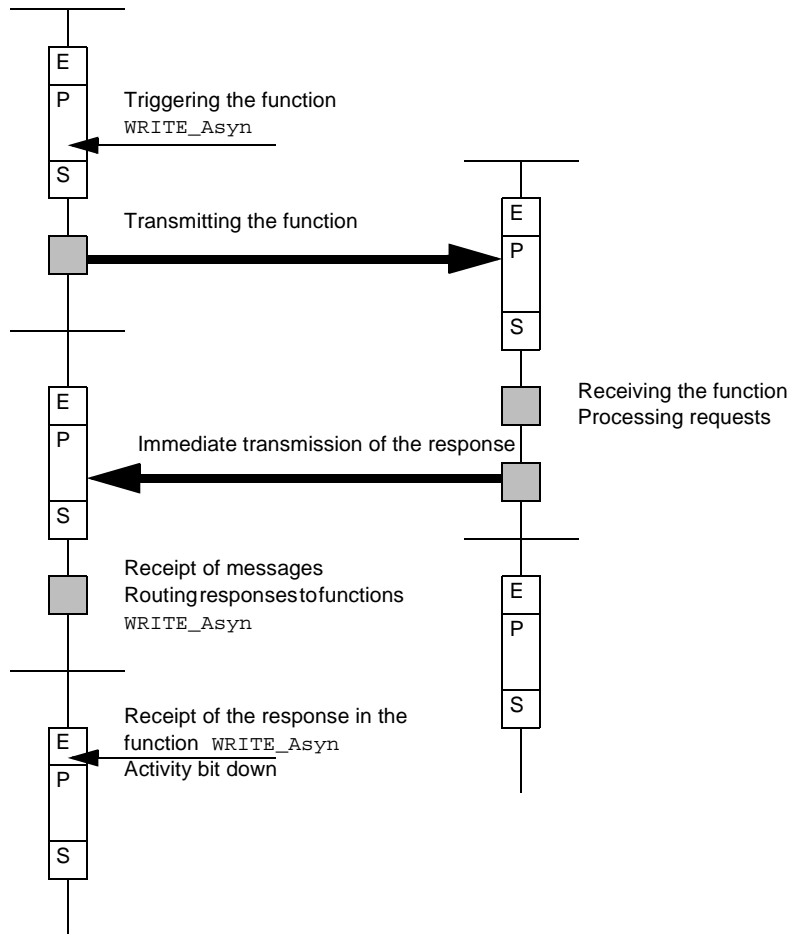
The size of the transmitting and reception buffers is expressed in words. It is 512 words or 1024 bytes.

<p><b>Note:</b> The asynchronous server function supports the UNI-TE V2.0 or V1 protocol. Functions <code>WRITE_Asyn</code> and <code>READ_Asyn</code> use the UNI-TE V2.0 protocol.</p>
--

---

**Principal of the exchanges**

The following diagram shows the exchanges between two stations for a function `WRITE_Asyn` :





**Syntax**

The syntax of communication functions `WRITE_Asyn` and `READ_Asyn` are presented in the following format :

```
WRITE_Asyn(ADR#{1.2}SYS, '%MW', 100, 800, %MW10:800,
%Mw1000:4)
READ_Asyn(ADR#{1.2}SYS, '%MW', 100, 800, %MW10:800, %MW1000:4)
```

The following table describes the different parameters of the function.

Parameter	Description
ADR#{1.2}SYS	Address of the exchange destination. This three word table stores the addresses of the type: {Network.Station} RackModule.Channel Slave or SYS.
'%MW'	String of characters specifying the type of object to be read : <ul style="list-style-type: none"> <li>● %M : internal bit</li> <li>● %MW : internal word</li> </ul>
100	Double word indicating index of the first object to be written or read.
800	Word specifying the number of objects to be written or read.
%MW10:800	Transmission buffer or receipt buffer. This table contains the values of the words to be written or the word read.
%MW1000:4	Management parameters. The operation report takes one of the following values: <ul style="list-style-type: none"> <li>● 16#00 : correct exchange</li> <li>● 16#01 : timeout, the response has not been able to be sent in less than 2 seconds</li> <li>● 16#02 : stop on user request (STOP,S0,INIT,a cold or warm restart)</li> <li>● 16#03 : Incorrect address format</li> <li>● 16#05 : incorrect management parameters</li> <li>● 16#07 : destination absent</li> <li>● 16#09 : size of reception buffer is insufficient</li> <li>● 16#10 : size of emission buffer is insufficient</li> <li>● 16#11 : absence of resource system ( 8 functions already active)</li> <li>● 16#19 : incorrect exchange number</li> <li>● 16#FF : message refused</li> </ul>

**Note:** It is necessary to program a Timeout value to stop an exchange that is in action when the response does not come back to the transmitter.

## 3.4 Characteristics of communication

---

### Presentation

**Subject of this sub- chapter** This sub-chapter presents the compatibility problems between the TSX Micro/Premium PLC and the series 7 PLC.

---

**What's in this section?** This section contains the following topics:

Topic	Page
Specifications for Communication between TSX Micro/Premium and Series 7	147
General rules for changing between applications	149
General rules for an exchange to a UNI-TE server	151
Other examples of compatibility	152

---

## Specifications for Communication between TSX Micro/Premium and Series 7

### At a Glance

To enable communication between a TSX Micro or a TSX Premium PLC and series 7 PLCs, such as TSX 47-107 or TSX 17, compatibility must be established on the level of communication blocks and then on the exchange level.

**Note:** The TER port of Premium PLCs only supports 10 or 11 bit format.

### Compatibility of Communication Blocks

PL7-3 defines 5 types of text blocks. A text block is characterized by destination entities, while a function specializes in a type of operation.

Correspondence between text block and communication function takes into account two parameters:

- Type of destination
- Type of communication operation

	Text block	CPL	SYS	TER	TXT	TLG	Equivalent functions
<b>Local</b>	Uni-telway	X	-	-	-	-	SEND_REQ, READ_VAR, WRITE_VAR (2)
	FIPIO	X (1)	-	-	-	-	SEND_REQ, READ_VAR, WRITE_VAR (2)
	Terminal port	-	-	X	-	-	PRINT_CHAR, INPUT_CHAR, OUT_IN_CHAR
	Character mode	X	-	-	-	-	PRINT_CHAR, INPUT_CHAR, OUT_IN_CHAR
<b>Remote</b>	UNI-TE	-	X	-	-	-	SEND_REQ, READ_VAR, WRITE_VAR (2)
	Uni-telway	X	-	-	-	-	SEND_REQ, READ_VAR, WRITE_VAR (2)
	FIPIO	X (1)	-	-	-	-	SEND_REQ, READ_VAR, WRITE_VAR (2)
	Application	-	-	-	X	-	DATA_EXCH
	Telegram	-	-	-	-	X	SEND_TLG, RCV_TLG
<b>Legend:</b>							
X	Correspondence with functions						
-	No correspondence						
(1)	Only with the integrated FIPIO link						
(2)	SEND_REQ performs the same operations as READ_VAR and WRITE_VAR						

**Compatibility of  
Exchanges**

Communication functions, which allow communication with the TSX 17, TSX 47, TSX 47 or 1000 series PLCs.

---

## General rules for changing between applications

### Introduction

Function `DATA_EXCH` allows transmitting, receiving of data to a PL7 application. This kind of processing is carried out by the `TXT` text block in PL7-3.

During an exchange between TSX Micro or TSX Premium and a series 7 PLC, the text blocks and the communication functions are adapted according to the direction of the exchange :

- From a TSX Premium PLC to a series 7 PLC ,
- From a series 7 PLC to a TSX Premium PLC ,

### Exchange from a Premium to a series 7

The TSX Premium transmits a communication function to a series 7 PLC.

- From the TSX Premium's side, the communication function must have the following address :  
`ADR#{Network.Station}APP.i` with `i`= number of text block receiver.
- From the side of the series 7 PLC, the text block must be initialized in the following way :  
`TXT, T` to the value `16#FF`.

**Note:** Parameter `TXT, A` contains the address of the transmitter (`Network.station`).

### Exchange from a series 7 to a Premium

The series 7 PLC transmits a text block to a TSX Premium PLC.

- From the TSX Premium's side, the communication function must have the following address :  
`ADR#{Network.Station}APP.i` with `i`= number of text block transmitter.
- From the side of the series 7 PLC, the text block must be initialized in the following way :  
`TXT, T` to the value `16#00`.

**Note:** Parameter `TXT, A` contains the destination address (`Network.station`).

## Examples

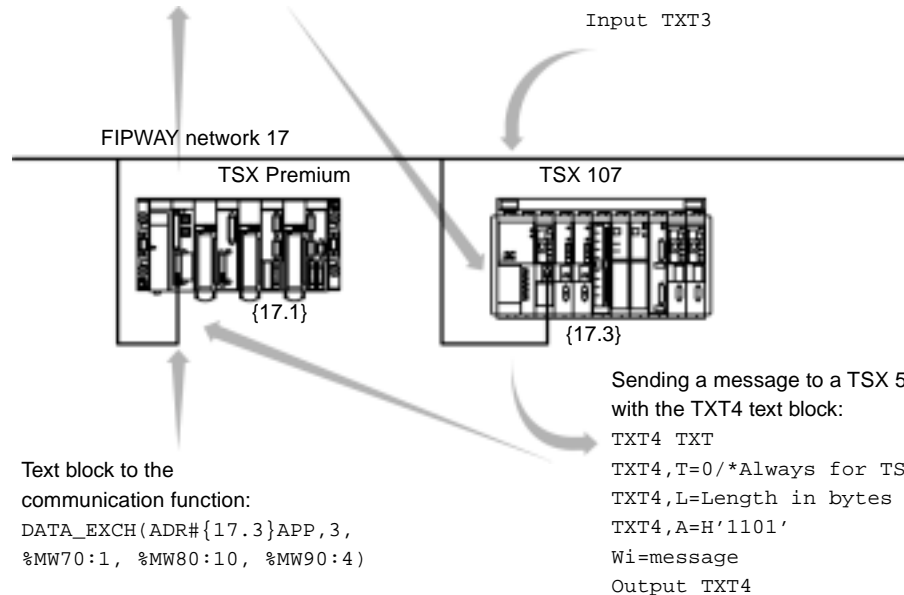
The examples handle the exchanges from a TSX Premium to a TSX 107 and vice versa. Value H'1101 corresponds to network 17 (H'11') and station (H'01').

Communication function  
to text block:

```
DATA_EXCH(ADR#{17.3}APP.3,2,
%MW20:10, %MW30:1, %MW50:4)
```

Receipt of a message from a TSX 57  
to a TXT3 text block:

```
TXT3 TXT
TXT3,=H'FF'/*Always FF*/
TXT3,L=Length
TXT3,A=h'1101'
Input TXT3
```



Text block to the  
communication function:

```
DATA_EXCH(ADR#{17.3}APP.3,
%MW70:1, %MW80:10, %MW90:4)
```

Sending a message to a TSX 57  
with the TXT4 text block:

```
TXT4 TXT
TXT4,T=0/*Always for TSX 37_57*/
TXT4,L=Length in bytes
TXT4,A=H'1101'
Wi=message
Output TXT4
```

## General rules for an exchange to a UNI-TE server

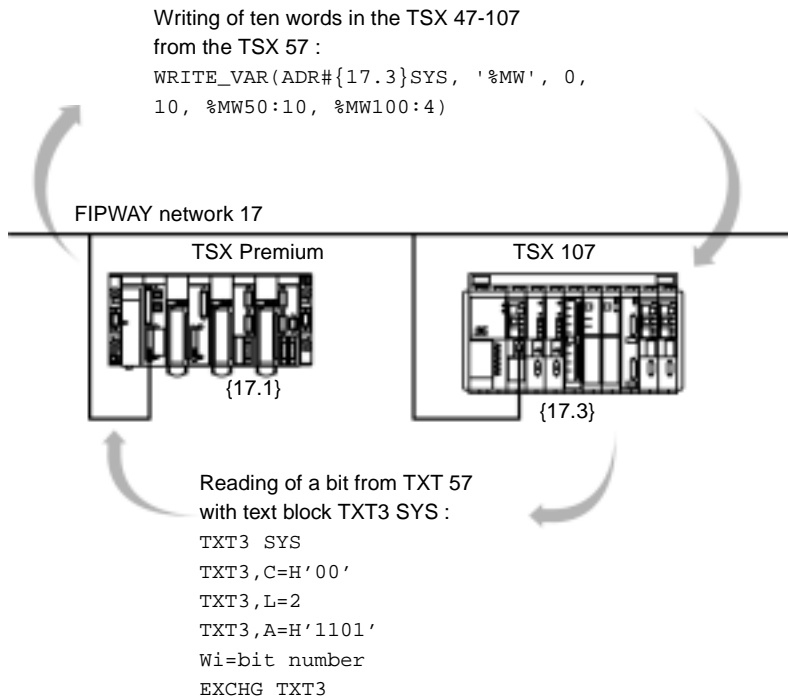
### Introduction

Function ( READ\_VAR , WRITE\_VAR , SEND\_REQ can communicate with the system of a TSX 47's server.

In the same way, a SYS Text Block also allows communication to a TSX Micro server and TSX Premium.

### Example

The examples handle the exchanges from a TSX Premium to a TSX 107 and vice versa. Value H'1101 corresponds to network 17 (H'11') and station (H'01').



## Other examples of compatibility

### Examples

To ensure compatibility, the following are the programming rules which should be respected :

<b>TSX 37/57 network 2, station 1</b>	<b>TSX 47 -107 network 2, station 2</b>	<b>Comment</b>
Transmission function DATA_EXCH(ADR#{2.2}APP, 8,2, %MW10:20, %MW50:1, %MW100:4) Transmission to text block 8 Initialization of length before transmission	Text block receipt TXT8,A=H'0201' / *transmitter address*/ TXT8,T=H'FF' INPUT TXT8 TXT8,L=40 Text block 8 can not recognize the transmitter number and so is placed in the mean while on all possible numbers ( 255)	There is therefore exchange compatibility between a transmitting function and a text block receiver.
Receipt function DATA_EXCH(ADR#{2.2}APP, 8,3, %MW10:1, %MW50:10, %MW100:4) Function awaiting a text block with any number	Text block transmission TXT8,A=H'0201' / *destination address*/ TXT8,T=H'00' TXT8,L=20 OUTPUT TXT8 Text block 8 always transmits to number 0 to communicate with a communication function	There is therefore exchange compatibility between a transmitting text block and a receiver function.
Function in exchange DATA_EXCH(ADR#{2.2}APP, 8,1, %MW10:1, %MW50:10, %MW100:4) The function transmits an outgoing message from text block 8 and begins to await a response from this same text block.	Text block transmission TXT8,A=H'0201' / *transmitter address*/ TXT8,T=H'FF' INPUT TXT8 TXT8,A=H'0201' / *destination address*/ TXT8,T=H'00' TXT8,L=20 OUTPUT TXT8 Text block 8 behaves identically like the last two examples	Therefore there is exchange compatibility between a function in exchange and a text block receiving and then transmitting.



<b>TSX 37/57 network 2, station 1</b>	<b>TSX 47 -107 network 2, station 2</b>	<b>Comment</b>
Function receiving and transmitting DATA_EXCH(ADR#{2.2}APP, 8,3, %MW10:1, %MW50:10, %MW100:4) DATA_EXCH(ADR#{2.2}APP, 8,2, %MW10:20, %MW50:1, %MW100:4)	Text block transmission/receipt TXT8,A=H'0201' / *destination address*/ TXT8,T=H'00' TXT8,L=20 Text block 8 can not recognize the number of the function, it is therefore impossible to initialize the TXT,T parameter to a value that can be used for transmitting and then for the receipt of the communication.	A text block in exchange and a function transmitting and then receiving are incompatible to do an exchange.

# 3.5                    Objects linked to communication

---

## Introduction

<b>Subject of sub-chapter</b>	This chapter presents the exchange modes of language objects linked to communication.
-------------------------------	---

---

<b>What's in this section?</b>	This section contains the following topics:								
	<table><tr><th>Topic</th><th>Page</th></tr><tr><td>Default exchanges</td><td>155</td></tr><tr><td>Specified exchanges : General points</td><td>157</td></tr><tr><td>Exchange and report management</td><td>159</td></tr></table>	Topic	Page	Default exchanges	155	Specified exchanges : General points	157	Exchange and report management	159
Topic	Page								
Default exchanges	155								
Specified exchanges : General points	157								
Exchange and report management	159								

---

## Default exchanges

### Introduction

An integrated interface where the addition of a module automatically expands the application of language objects allowing the programming of this interface or this module.

These objects correspond to the images of the input/output of the module or of the integrated interface.

The bits %I and the words %IW, images of the input values of the module, are automatically updated in the PLC processor at the beginning of the task, whether the task is in RUN or in STOP mode.

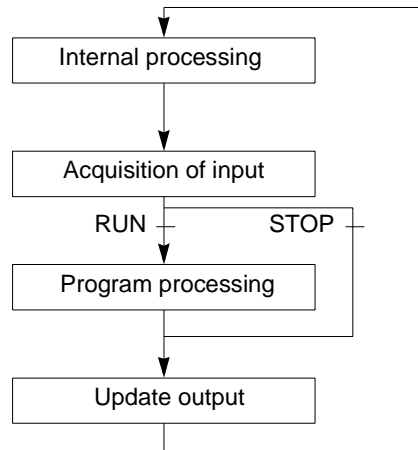
The bits %Q and the words %QW, images of the output values of the module, are automatically updated in the module by the processor at the end of the task, the task being in RUN mode.

**Note:** When the task is in STOP mode, following the chosen configuration :

- the output is put into fallback position (fallback mode),
- the output is maintained at its last value (maintenance mode).

### Illustration

The chart illustrates the operation cycle relating to a PLC task (cyclic execution).



**Examples**            The table below shows some examples of default exchanges relating to a given operation.

Object	Operation	Description
%I103.1	TOR	Gives status of channel 1 of the module situated in position 3 on rack 1.
%IW4.2	Analog	Gives the analog value of channel 2 of the module situated in position 4 on rack 0.
%IW0.1.1:x0	Uni-telway	Gives the general state of the slaves, the communication channel is situated in channel 1 of the processor in position 0 of rack 0.
%Q0.2.1\0.8	FIPIO	Gives the status of output 8 from slave 1 on FIPIO bus.
%I6.5.ERR	-	Indicates, when the bit is set to 1, that channel 5 of the module situated in position 6 of rack 0 has an error.
%I107.MOD.ERR	-	Indicates, when the bit is set to 1, that the module situated in position 7 of rack 1 has an error.

---

## Specified exchanges : General points

---

### Introduction

The specified exchanges are the exchanges carried out on request of the program user with help instructions :

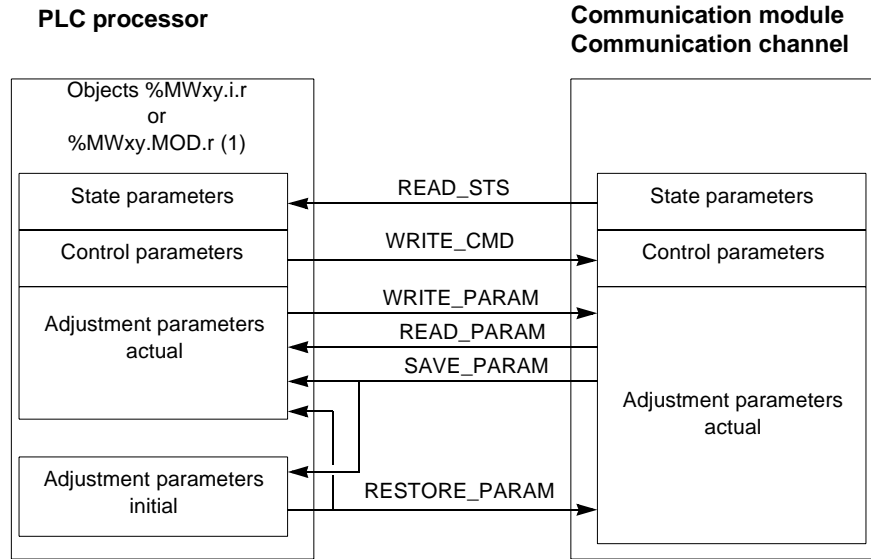
- READ\_STS :
- WRITE\_CMD : ,
- WRITE\_PARAM : ,
- READ\_PARAM : ,
- SAVE\_PARAM : ,
- RESTORE\_PARAM : .

These exchanges apply to the whole of %MW objects which are of the same kind (state, controls or parameters) on the same channel.

**Note:** These objects are not necessary when programming of a task function, but they carry extra information (ex : Terminal block fault, module absent...) and the extra orders to implement an advanced programming of the task functions (for more information on the specified exchange that is specific to a task, refer to the chapter that deals with this subject).

### General principles on the use of the specified instructions

The diagram below presents the different kinds of specified exchanges that are possible between the PLC's processor and the module (or the built-in interface ).



(1) Only with the instructions READ\_STS and WRITE\_CMD.

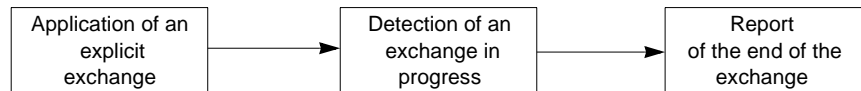
### Management of the exchanges

During a specified exchange, it may be interesting to monitor the progress that this makes, in order to, for example, not take into consideration the data read until the exchange has effectively been carried out.

For this, two kinds of information are available :

- Detecting an exchange whilst in action :
- the end of exchange report.

The diagram below describes the principal of the management of an exchange



### Logic channel %Chxy.i

The channel %CHxy.i is a general syntax to update all the associated objects to this channel via specified instructions.

**Example** READ\_STS%CH102.3

## Exchange and report management

### Introduction

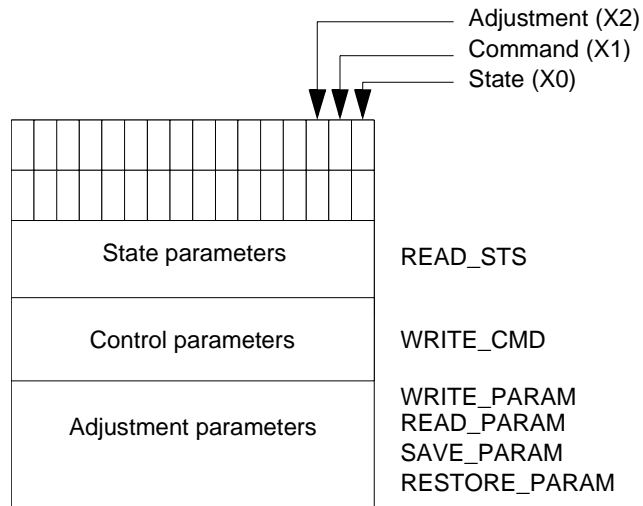
When the data is exchanged between the PLC memory and the module, the considerations taken by the coupler may require the running of several task cycles. To manage the exchanges, 2 words are used :

- %MWxy.i.0 : Exchange in progress,
- %MWxy.i.1 : Report

**Note:** These words are the object of a detailed description in each task insert.

### Illustration

The illustration below presents the different data bits for the management of the exchange.



**Description of the data bits**

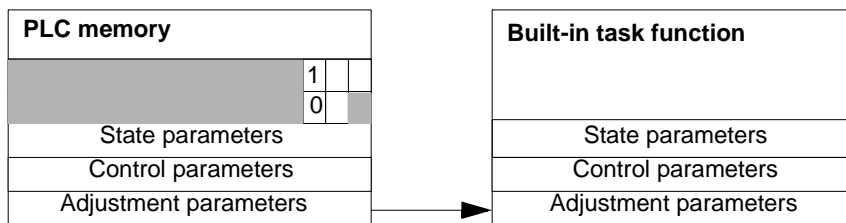
Each of the word bits %MWxy.i and %Mwxy.i.1 is associated to a kind of parameter :

- the bits in position 0 are associated to the state parameters :
  - the bit %Mwxy.i.X0 indicates if a request to read the status word is in action,
- the bits in position 1 are associated to the control parameters :
  - the bit %Mwxy.i.0:X1 if the control parameters are sent to the communication channel,
  - the bit %MWxy.i.1:X1 if the control parameters are accepted by the communication channel,
- the bits in position 2 are associated to the adjustment parameters :
  - the bit MWxy.i.0:X2 indicates if the adjustment parameters are exchanged with the communication channel ( by WRITE\_PARAM,READ\_PARAM,SAVE\_PARAM,RESTORE\_PARAM),
  - the bit %MWxy.i.1:X2 specifies if the adjustment parameters are accepted by the communication channel. If the exchange has progressed correctly, the bit changes to 0,

**Note:** The exchange and report words also exist at the level of module (%MWxy.MOD and %MWxy.MOD.1).

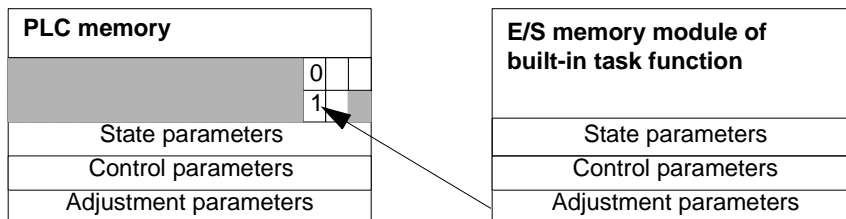
**Example**

Phase 1 : Transmitting data with the help of the instruction WRITE\_PARAM



When the instruction is scanned by the PLC processor, the bit **Exchange in progress** is set to 1 in %MWxy.i.0:X2

Phase 2 : Analysis of data by the E/S module and the report





When the data is exchanged between the PLC memory and the module, the considerations taken by the coupler is managed by the bit %MWxy.i.1:X2 : Report (0 =correct exchange, 1= unsuccessful exchange).

<b>Note:</b> There are not any adjustment parameters at the modular level.
--

---



---

# Configuration of the operations function Communication



---

## Presentation

**Subject of this chapter** This chapter presents the configuration and the debugging of the operations function Communication.

**What's in this chapter?** This chapter contains the following topics:

Topic	Page
Configuration of the communication function	164
Reminders concerning the configuration editor	165
How to declare a communication module	166
How to declare a communication channel in a processor or module TSX SCY 21600/21601	167
Description of configuration screens for communication	168
Description of communication debugging screens	170

## Configuration of the communication function

---

### Introduction

Before creating an application program, it is necessary to define the physical operating context in which it will be ran, this means the type of processor and the modules positioned in each slot.

The use of communication functions also requires the definition of the parameters of the communication channels being used ( choice of protocol, definition of the specific parameters,...).

To do this the PL7 Micro,PL7 Junior and PL7 Pro software propose the configuration tool which allows these operations to be carried out easily.

In connected operation, they also propose a debugging function that allows certain parameters to be adjusted in order to be better adapted to the application.

---

## Reminders concerning the configuration editor

### Presentation

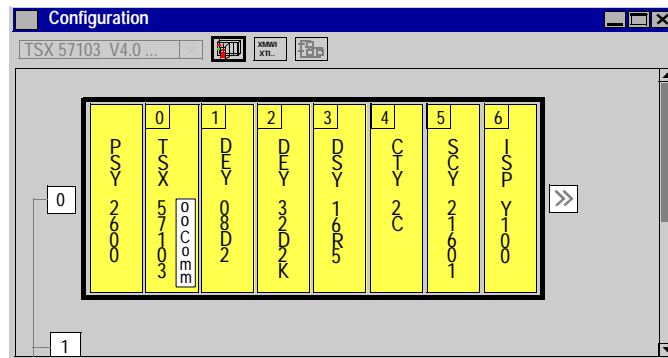
The configuration editor enables the various constitutive elements of the PLC to be declared and configured in a graphical and intuitive manner:

- rack,
- power supply,
- processor,
- application-specific modules.

In online mode the configuration editor also performs debug, adjustment and diagnostics functions.

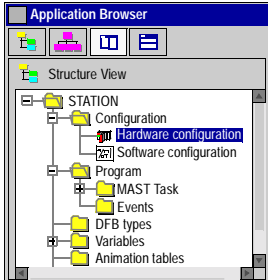
### Illustration

The following screen provides an example of a hardware configuration.



### Accessing the editor

The following table shows the different ways of accessing the configuration editor.

From:	Action
the menu bar	Select <b>Tools</b> → <b>Configuration</b> .
the application browser	Double-click on <b>Hardware configuration</b> or select it using the arrow keys and confirm with <b>Enter</b> . 

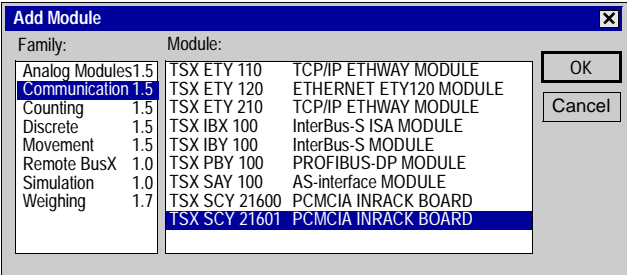
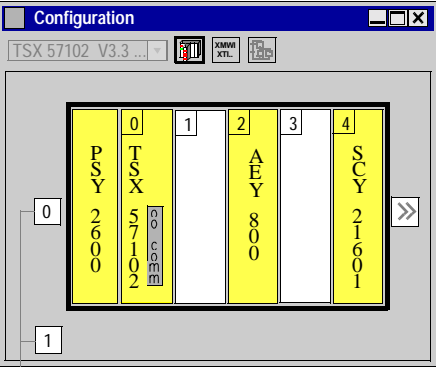
## How to declare a communication module

### Introduction

This operation allows the declaration of a communication module in a PLC rack through the use of software.

### Steps to follow

The procedure is as follows :

Step	Action
1	<p>Double click on the slot in which the communication module must be configured.</p> <p><b>Result</b></p> <p>Screen <b>Add a module</b> appears.</p> 
2	Select in the field <b>Label</b> the choice <b>Communication</b>
3	Select in the field <b>Module</b> the reference of the module.
4	<p>Validate the choice with <b>Ok</b>.</p> <p><b>Result</b></p> <p>The module is declared in its slot. The latter is greyed out and contains the reference of the module.</p> 

---

## How to declare a communication channel in a processor or module TSX SCY 21600/21601

---

### Introduction

This operation allows the declaration of a communication channel in a processor or in a module TSX SCY 21600/T21601 through the use of software.

When declaring a communication channel in a module TSX SCY 21600/21601, it is first necessary to declare this communication module in the PLC rack.

---

### Steps to follow

The procedure is as follows:

Step	Action
1	To which element does the communication channel belong? <ul style="list-style-type: none"><li>• if the processor: go on to step 2.</li><li>• if module TSX SCY 21600/21601 : go on to step 3.</li></ul>
2	<ul style="list-style-type: none"><li>• select in the processor's slot the channel <b>COMM</b>.</li><li>• select the command <b>Services</b> → <b>Open the module</b></li></ul>
3	<ul style="list-style-type: none"><li>• select the module <b>SCY 2160•</b>.</li><li>• select the command <b>Services</b> → <b>Open the module</b>.</li></ul>

---

## Description of configuration screens for communication

### Introduction

The configuration screen dedicated to communication operations is split into two distinct parts:

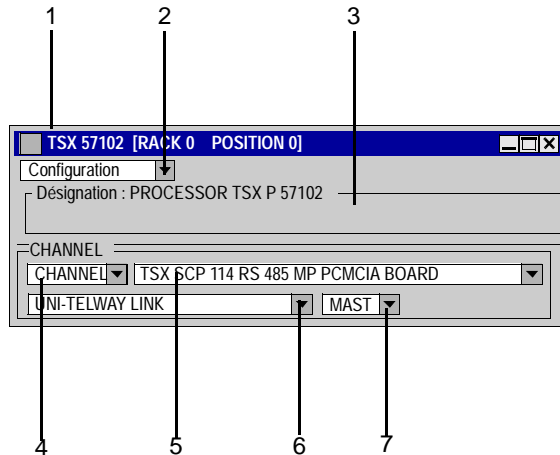
- the upper part, common to all types of configuration screens, is dedicated to information at module level and at communication channel level. Its contents are detailed hereafter.
- the lower part of the screen is dedicated to configuration data and parameters. This zone, which is specific to the type of communication chosen, is detailed in the supplements to this document related to the different types of communication.

### Configuration rules

It is forbidden to modify the communication channel configuration parameters while in connect mode (PLC in RUN mode).

### Illustration

This zone gives access to the visualization and modification of the communication channel parameters in local mode.





**Description**      The table below presents the different elements of the communication channel and their functions.

Address	Element	Function
1	Title bar	This indicates the commercial reference and the position of the module
2	Choosing the function	This drop-down list allows the choice of mode: <ul style="list-style-type: none"> <li>● configuration,</li> <li>● debug (only when connected).</li> </ul>
3	Module zone	This zone displays the designation of the selected module.
4	Communication channel	This drop-down list allows the choice of the communication channel: <ul style="list-style-type: none"> <li>● channel 0 corresponds to the terminal port.</li> <li>● channel 1 corresponds to the PCMCIA card slot.</li> </ul>
5	Communication module	This drop-down list allows the selection of the communication module assigned to the channel
6	Protocol	This drop-down list allows the selection of the communication protocol.
7	Task	This drop-down list allows the assigning of the communication module to a PLC task. <b>Note :</b> The communication modules must always be declared in MAST task).

## Description of communication debugging screens

### Introduction

The debugging screen dedicated to communication operations is split into two distinct parts:

- the upper part, common to all types of debugging screens, is dedicated to information at module level and at communication channel level. Its contents are detailed hereafter.
- the lower part of the screen is dedicated to debugging data and parameters. This zone, which is specific to the type of communication chosen, is detailed in the supplements to this document related to the different types of communication.

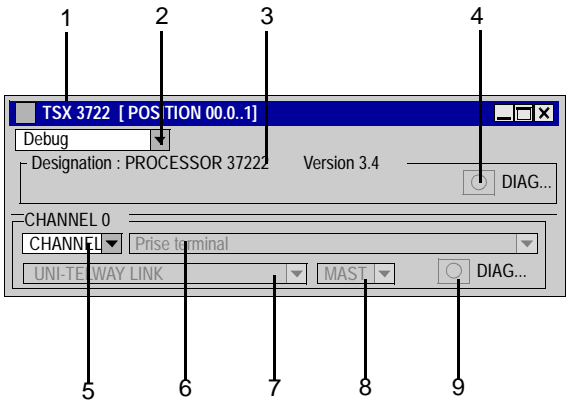
### How to access the screen

Access to debug mode is only possible when in connected mode.

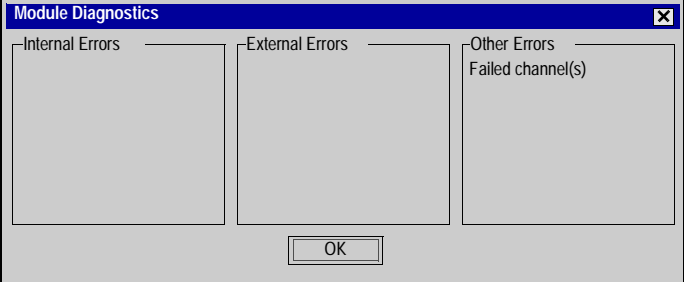
Step	Action
1	Access the configuration screen.
2	Select the mode <b>Debug</b> in the list of function options.

### Illustration

This zone gives access to the communication channel diagnostics.



**Description** The table below presents the different elements of the communication channel and their functions.

Address	Element	Function
1	Title bar	This indicates the commercial reference and the position of the module
2	Choosing the function	This drop-down list allows the choice of mode: <ul style="list-style-type: none"> <li>● configuration,</li> <li>● debug (only when connected).</li> </ul>
3	Module zone	This zone displays the designation of the selected module. In addition for certain types of modules, this zone displays indicators which allow you to see the state of the module.
4	Diagnostic button	<p>When an error is detected at module level, a button <b>DIAG</b> allows access to status information for this module (this button is disabled or enabled according to the value of the module's status bit : %I4.0.MOD.ERR).</p> 
5	Communication channel	This drop-down list allows the choice of the communication channel: <ul style="list-style-type: none"> <li>● channel 0 corresponds to the terminal port.</li> <li>● channel 1 corresponds to the PCMCIA card slot.</li> </ul>
6	Communication module	This drop-down list displays the selection of the communication module assigned to the channel
7	Protocol	This drop-down list displays the selection of the communication protocol.
8	Task	<p>This drop-down list displays the PLC task assigned to the communication module.</p> <p><b>Note :</b> The communication modules must always be declared in MAST task).</p>

Address	Element	Function
9	Diagnostic button	<div><p>When an error is detected at channel level, a button <b>DIAG</b> allows access to status information for this channel (this button is disabled or enabled according to the value of the channel's status bit : %I4.0.MOD.ERR).</p><div><div>Channel Diagnostics</div><div><div>Internal Errors</div><div>External Errors</div><div>Other Errors</div></div><div><div>No device available on the channel</div><div>A device in an error condition</div></div><div>OK</div></div></div>

---

# Remoting of Nano PLCs



---

## Introduction

### Subject of this part

This part introduces the principles of configuring and remoting TSX Nano PLCs via PL7 software.

### What's in this part?

This part contains the following chapters:

Chapter	Chaptername	Page
5	General	175
6	Nano PLC remoting services	185
7	Configuring remoting of Nano PLCs	197
8	Programming Nano PLC remoting	203
9	Debugging Nano PLC remoting	209
10	Language objects associated to Nano PLC remoting	213



Introduction

Subject of Chapter

This Chapter introduces the remoting of TSX Nano PLCs and their services.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Introduction	176
Compatibility	177
Performances: Network Cycle Time:	178
Performance: positioning an output	179
Operating mode	182

## Introduction

---

### Introduction

This type of link is used to connect a TSX Micro PLC to TSX Nano PLCs to use them as input/output buses.

The link offers the following different services:

- input/output data exchange,
- application data exchange.

### Associated manuals

If you require further information you should consult the following manuals:

Title	Description
TSX Micro PLCs - Installation manual	Hardware installation



## Compatibility

### Hardware

This type of communication is available only with the TSX STZ 10 module for the following PLCs:

- TSX 37 10
- TSX 37 20
- TSX 37 21

**Note:** TSX 37 05 and TSX 37 08 PLCs cannot accept the TSX STZ 10 module.

Using this module, it is possible to connect up to a maximum of four of the following elements:

- TSX 07 20•/21•/30•/31• PLCs used as:
  - remoted input/output blocks (maximum of 4 PLCs),
  - other PLCs (3 maximum).
- TSX AMN 4 •analog input/output modules (maximum of 3 modules).
- TSX EX 07 •input/output extension modules (maximum of 1 module).


On a remoted link, address 1 must be either left free or taken up by an I/O block (TSX 07 EX•or TSX 07 20•/21•/30•/31•used as remoted inputs/outputs).

### Software

While configuring the link, TSX AMN 4• analog modules must be declared as slave PLCs.

It is vital that module TSX STZ 10 is installed and configured in slot 4 of the TSX Micro PLC.

You are advised, while configuring the module, to associate it with the **MAST** (master) task for performance reasons.

	<b>DANGER</b>
	<p>When the PLC is in STOP, the output value of TSX AMN 4• analog modules is maintained contrary to other input/output which switch to their configured state.</p> <p><b>Failure to observe this precaution will result in death or serious injury.</b></p>

## Performances: Network Cycle Time:

---

### Definition

The network cycle time corresponds to the processing time of all the devices present on the link, i.e.:

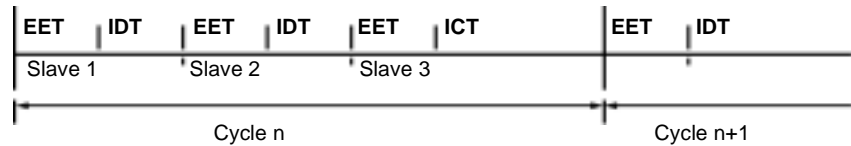
- the updating of the inputs / outputs for an input/output type device,
- the application data processing for an AP slave type device.

The length of the network cycle depends upon:

- the number and type of the slave devices,
- the line speed.

### Example

The following example deals with a network cycle with 3 devices on the



EET link. The elementary exchange time is the time required for data to be exchanged between a master and a slave.

IDT. Inter-device time is the processing time required between two exchanges with slaves.

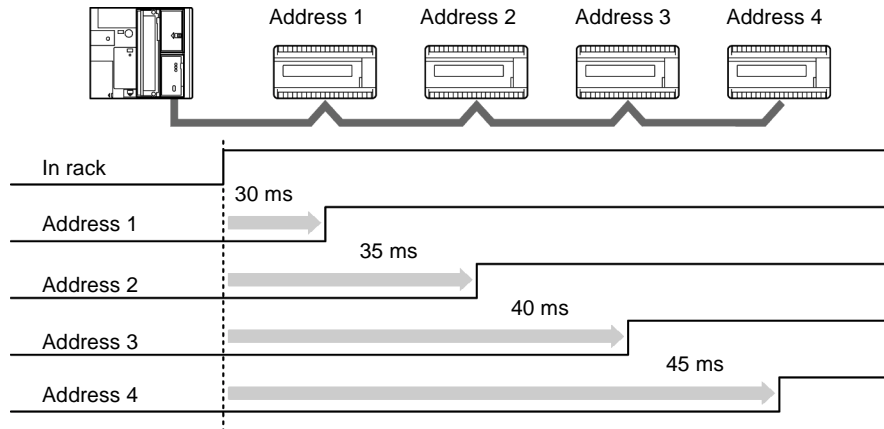
ICT. Inter-cycle time is the processing time required between two cycles.

---

## Performance: positioning an output

### Introduction

The diagram below shows the time gap between positioning an output in the rack and positioning a remote output (in theory, the network cycle time is less than the PLC cycle time, and the transmission speed is 38400 bits/s):



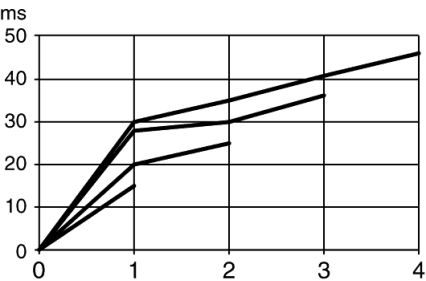
If several devices are present, the one with the lowest address has the shortest positioning time.

The times given here are the maximum values that take into account the link cycle times and the processing times of the remote devices. These times are asynchronous in relation to each other

The following times are given for a transmission speed of 38400 bits/s:

Number of slaves	Address 1	Address 2	Address 3	Address 4
1	15 ms	-	-	-
2	20 ms	25 ms	-	-
3	27 ms	30 ms	36 ms	-
4	30 ms	35 ms	41 ms	46 ms

Associated graph

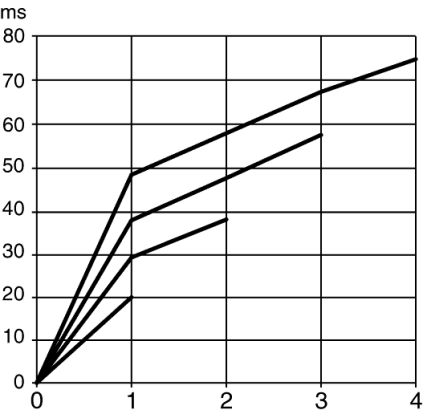


**For a speed of 19200 bits/s**

The following times are given for a transmission speed of 19200 bits/s:

Number of slaves	Address 1	Address 2	Address 3	Address 4
1	20 ms	-	-	-
2	29 ms	38 ms	-	-
3	38 ms	47 ms	57 ms	-
4	48 ms	57 ms	66 ms	75 ms

Associated graph



**For a speed of  
9600 bits/s**

The following times are given for a transmission speed of 9600 bits/s:

Number of slaves	Address 1	Address 2	Address 3	Address 4
1	32 ms	-	-	-
2	47 ms	64 ms	-	-
3	66 ms	83 ms	100 ms	-
4	84 ms	101 ms	118 ms	136 ms

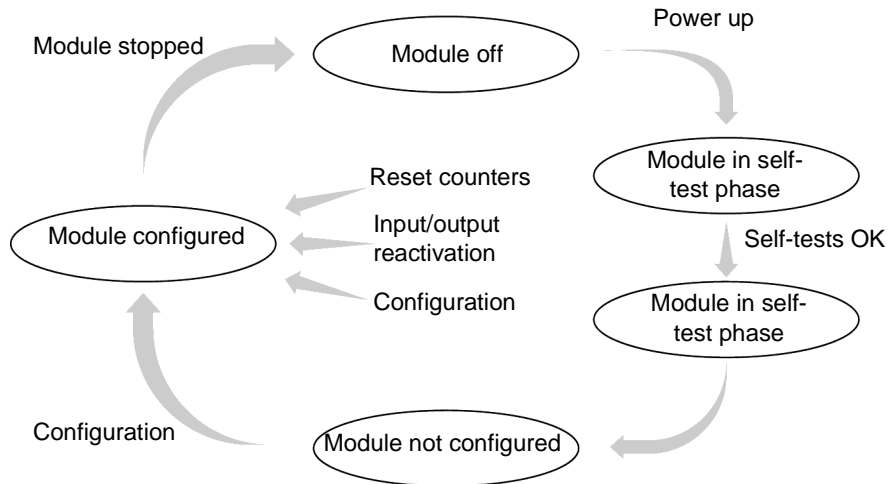
Associated graph



## Operating mode

**Introduction** The following graph describes operating modes for the TSX STZ 10 module.

**General graph** The operating mode is as follows:



## Operation

- after power up, the module performs self-testing. It then initializes the application configuration with the configuration physically present (learning phase), and selects the default speed of 19200 bits/s.
- if there is no PL7 application in the PLC, the module is loaded with the configuration that is actually present, and then communicates with input/output devices only.
- the configuration phase, triggered by the PLC, is used to take into account the application configuration (speed, configured devices, etc.).
- if there is a PL7 application in the PLC, the application configuration is sent to the module. In the event of inconsistency or if one of the devices is faulty, the ERR LED flashes. Inconsistent devices are periodically interrogated by the PLC. This resets th
- when the PLC is in STOP, the module exchanges data only with consistent input/output blocks. When the PLC switches to RUN, the module also starts to exchange data with consistent slave PLC devices. Common word exchanges with remote TSX 07 PLCs are inhibited w

- in case of power outage, the PLC processor performs a warm restart and automatically reconfigures the module.
  - in case of a communication break with the PLC processor, the module stops all exchanges on the bus.
-





---

## Introduction

**Subject of Chapter**

This Chapter introduces the services supported by TSX Nano PLC remoting and its services.

**What's in this chapter?**

This chapter contains the following topics:

Topic	Page
Exchanging input/output data	186
Exchanging application data	187
Exchanging data with an analog module	189
Contents of %QW words in writing	190
Contents of %IW words on reading	192
Conversion of analog values of input channels	194
Mixed link	196

---

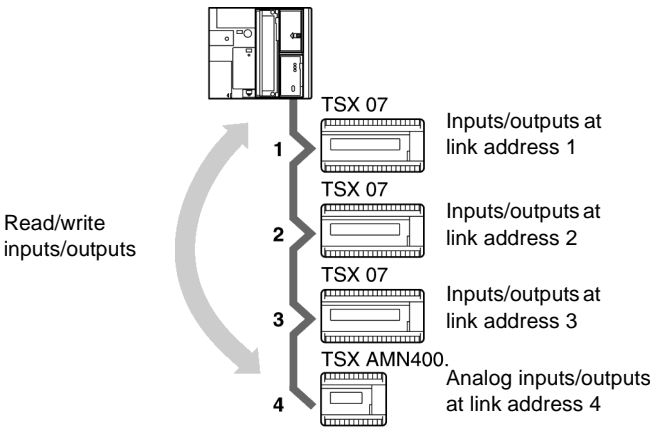
## Exchanging input/output data

### Introduction

This service provides direct access to inputs/outputs for discrete and analog TSX Nano PLCs.

### Exchanged objects

This module is used to read and write inputs and outputs:



### Addressing

Inputs/outputs can be accessed by using the following addressing:

- inputs: %I\4.0\link address.input position
- outputs: %Q\4.0\link address.output position

The following table shows the different addressing parameters:

Parameter	Description
%I or %Q	Type of object to be accessed (input or output).
\4.0\	<ul style="list-style-type: none"><li>● 4: module addresses in TSX Micro PLC.</li><li>● 0: channel address.</li></ul>
Link address	Address of connection point for destination PLC (from 1 to 4).
Input or output position	Destination PLC input or output number.

### Example

TSX Micro PLC:

- reads input 2 from PLC at link address 2: %I\4.0\2.2.
- writes output 4 from PLC at link address 3: %Q\4.0\3.4.

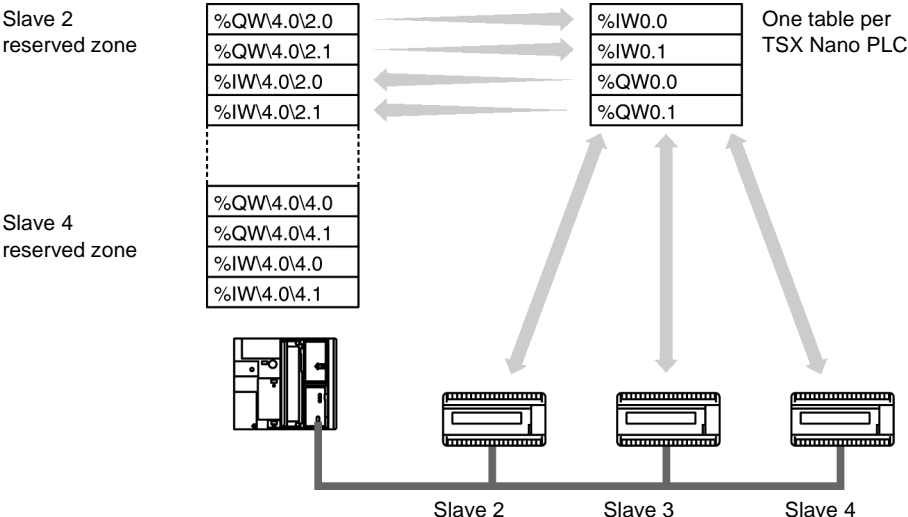
# Exchanging application data

## Introduction

This service is used to exchange application data between a TSX 37 PLC and up to three Nano PLCs. This data, which is limited to four words (two words being produced and two words being consumed) by the Nano PLC, can be exchanged in both directions.

## Exchanged objects

This module is used to exchange input words and output words:



**Addressing**

Input/output words can be accessed by using the following addressing:

- inputs: %IW\4.0\link address.word number
- outputs: %QW\4.0\link address.word number

The following table shows the different addressing parameters:

Parameter	Description
%IW or %QW	Type of object to be accessed (input or output word).
\4.0\	<ul style="list-style-type: none"><li>● 4: module address in TSX Micro PLC.</li><li>● 0: channel address.</li></ul>
Link address	Address of connection point for destination PLC (from 1 to 4).
Number of the word	Destination PLC input or output word number.

---

**Example**

TSX Micro PLC:

- reads 0 and 1 input words from the PLC at link address 2: %IW\4.0\2.0 and %IW\4.0\2.1.
  - writes 0 and 1 output words from the PLC at link address 4: %QW\4.0\3.0 and %QW\4.0\3.1.
-

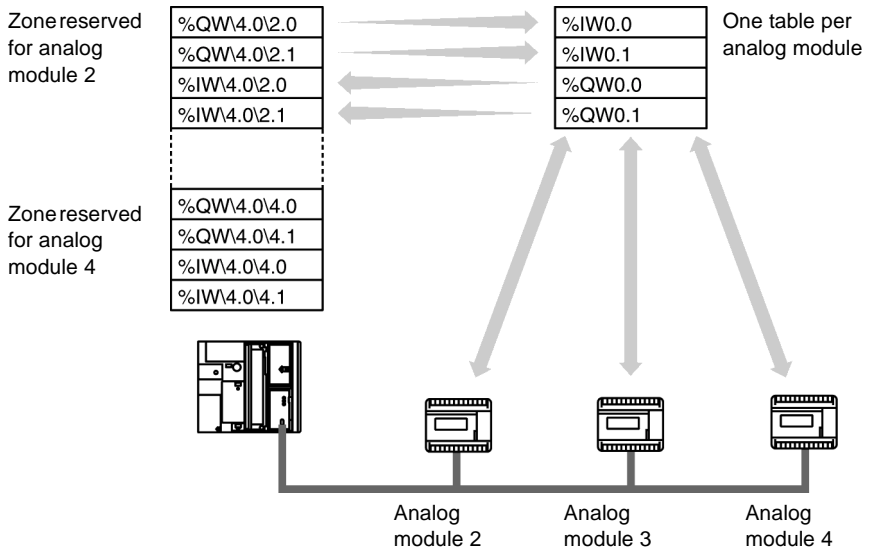
## Exchanging data with an analog module

### Introduction

Analog inputs/outputs addressing is identical to that of Nano PLC extensions.

### Exchanged objects

This module is used to exchange input words and output words:



An analog module is made up of 3 input channels and 1 output channel. Words %QW and %IW exchanged with these modules contain, for each channel:

- configuration parameters,
- values,
- status bits.

For input channel 1, resolution depends on the number of configured channels: 12 bits if channels 0 and 1 are configured, 8 bits if channels 0, 1, and 2 are configured.

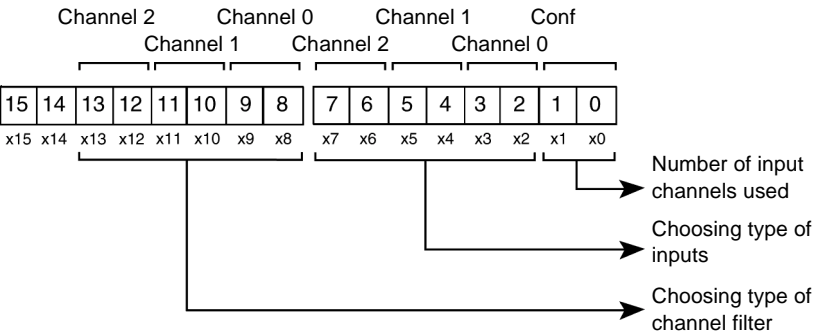
## Contents of %QW words in writing

### Introduction

Output words %QW\4.0i.0 and %QW\4.0i.1 contain configuration parameters for each input channel, and the analog value for the output channel.

### Contents of word %QW\4.0i.0

This word contains the configuration for analog input.



### Significance of word bits %QW\4.0i.0

Selection of the channels used

x1	x0	Number of input channels used
0	0	None
0	1	Channel 0
1	0	Channel 0 and channel 1
1	1	Channel 0, channel 1 and channel 2

Choice of input type (valid for pairs (x2,x3), (x4,x5), (x6,x7))

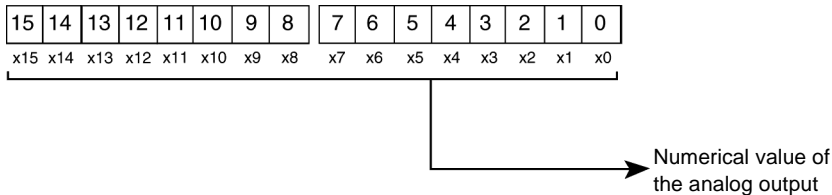
x3 x5 x7	x2 x4 x6	Type of inputs for the channel
0	0	10 V voltage input
0	1	0 10 V voltage input
1	0	10 20 mA current input
1	1	4 20 mA current input

Choice of filter type (valid for pairs (x8,x9), (x10,x11), (x12,x13))

<b>x9 x11 x13</b>	<b>x8 x10 x12</b>	<b>Type of filter for the channel</b>
0	0	Hardware filter
0	1	150 ms filter
1	0	750 ms filter
1	1	3 s filter

**Contents of word  
%QW4.0i.1**

This word contains the numerical value of the analog output.

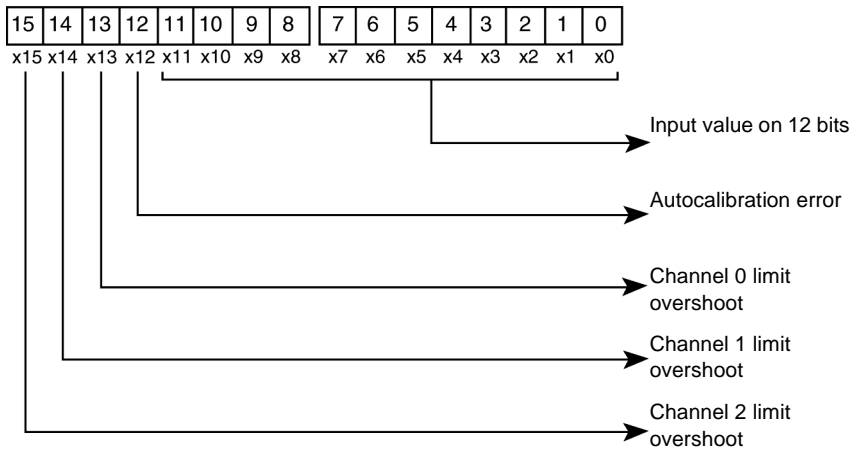


The numerical value of the output is defined on 15 bits. The last x15 bit is the sign bit.

## Contents of %IW words on reading

**Introduction** Input words %IW4.0*i*.0 and %IW4.0*i*.1 contain the status bits and the value of each channel.

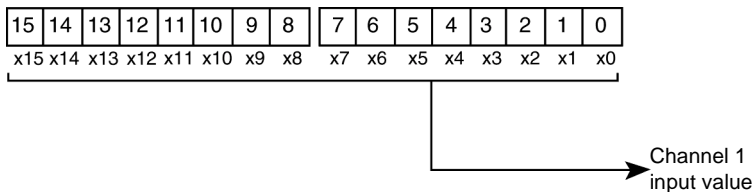
**Contents of word %IW4.0*i*.0** This word contains the value of the 0 analog input and the status of the module.



The analog input value is defined on 11 bits, and bit x11 is the sign bit. The other bits from x12 to x15 indicate the state of the module.

**Contents of word %IW4.0*i*.1** The contents of the second %IW4.0*i*.1 word depend on the configuration chosen for channels 0 and 1.

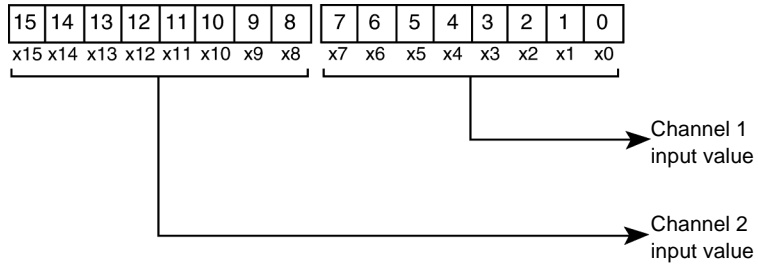
**Configuration 1:** inputs 0 and 1 are configured. The value of analog channel 1 is of the form



In this configuration, the value is defined on 15 bits, and the last x15 bit is the sign bit.



**Configuration 2:** all 0, 1 and 2 inputs are configured. The word %IW\4.0\i.1 contains the value of channel 1 and channel 2.



In this configuration, the channel values are defined on 7 bits. Bits x7 and x15 are the sign bits.

## Conversion of analog values of input channels

---

### Introduction

The input value read to be processed by PL7 must be defined on 16 bits, with bit x15 being the sign bit.

After being read, channel 0 must always be processed by a software application so that PL7 can use the result.

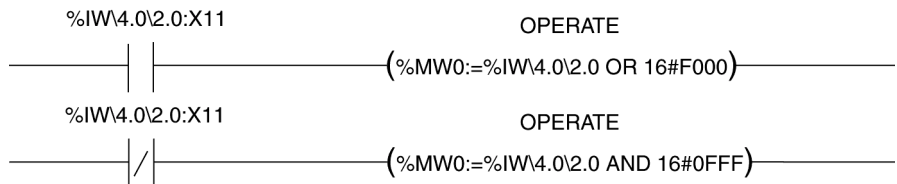
When channels 0 and 1 are configured, the value read on channel 1 does not need processing. When all channels (0, 1 and 2) are configured, the channel 1 and 2 readings must be processed.

For indication purposes, the following examples are given for an analog module connected to address 2.

---

### Converting channel 0 into Ladder language

%MW0 contains the value of channel 0 coded on 16 bits.



### Converting channel 0 into List language

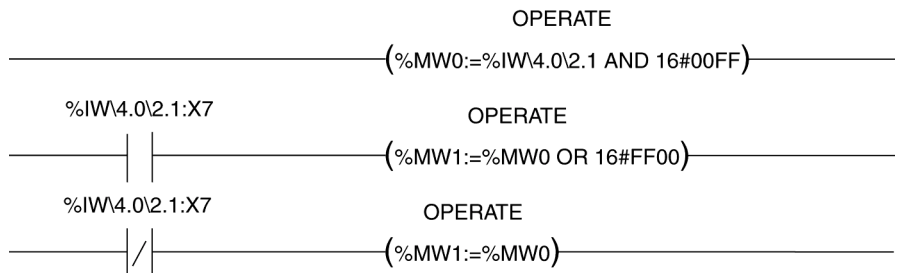
%MW0 contains the value of channel 0 coded on 16 bits.

```
LD      %IW\4.0\2.0:X11
[ %MW0 := %IW\4.0\2.0 OR 16#F000 ]
LDN     %IW\4.0\2.0:X11
[ %MW0 := %IW\4.0\2.0 AND 16#0FFF ]
```

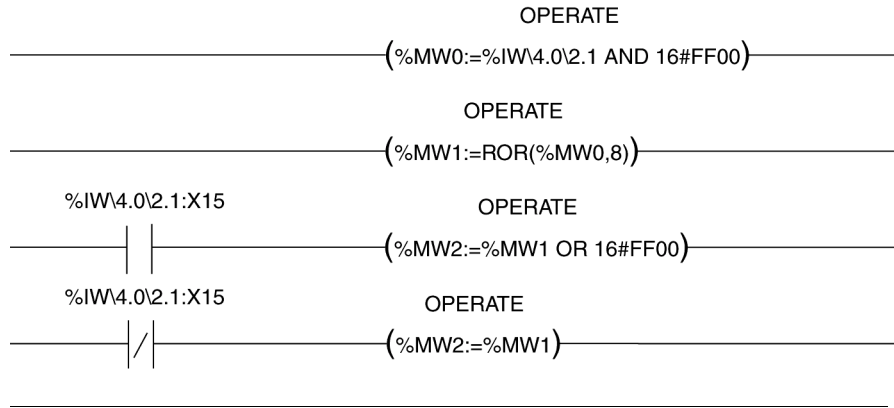
---

### Converting channels 1 and 2 into Ladder language

%MW1 contains the value of channel 1 coded on 16 bits.



%MW2 contains the value of channel 2 coded on 16 bits.



### Converting channels 1 and 2 into List language

%MW1 contains the value of channel 1 coded on 16 bits.

```
LD      1
[ %MW0 := %IW\4.0\2.1 AND 16#00FF ]
LD      %IW\4.0\2.1:X7
[ %MW1 := %MW0 OR 16#FF00 ]
LDN     %IW\4.0\2.1:X7
[ %MW1 := %MW0 ]
```

%MW2 contains the value of channel 2 coded on 16 bits.

```
LD      1
[ %MW0 := %IW\4.0\2.1 AND 16#FF00 ]
LD      1
[ %MW1 := ROR( %MW0, 8 ) ]
LD      %IW\4.0\2.1:X15
[ %MW2 := %MW1 OR 16#FF00 ]
LDN     %IW\4.0\2.1:X15
[ %MW2 := %MW1 ]
```

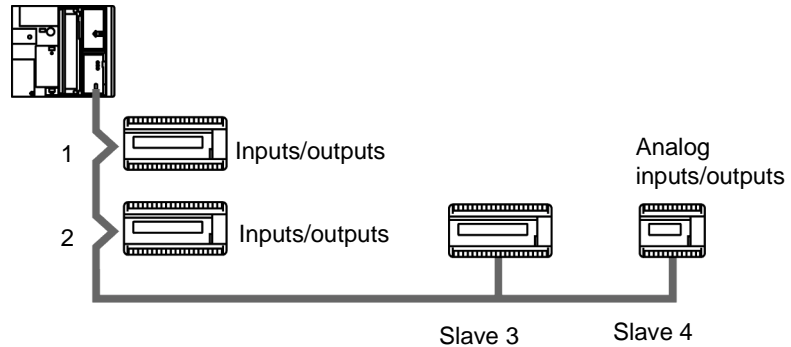
## Mixed link

---

### Introduction

It is possible to mix the input/output exchanges and the exchanges of application data on the same link.

TSX Micro master



**Note:** In this case, the first device must always be configured in a block of inputs/outputs.

---

# Configuring remoting of Nano PLCs



---

## Introduction

### Subject of this Chapter

This Chapter describes the Configuration process during set-up of TSX Nano PLC remoting.

### What's in this chapter?

This chapter contains the following topics:

Topic	Page
How to access parameters of module TSX STZ 10	198
Configuration screen for remoting Nano PLCs	199
Modbus parameters associated with the application	200

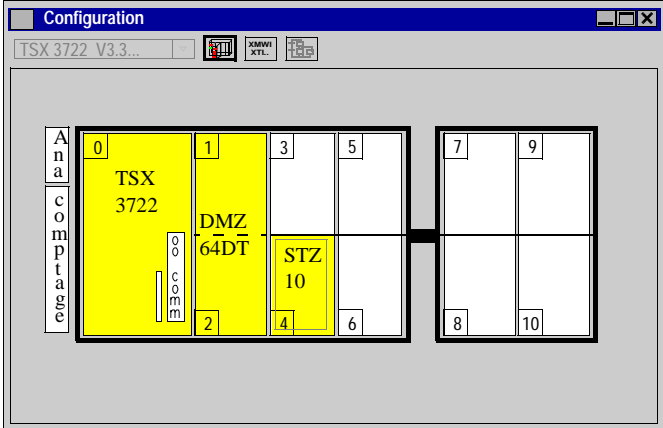
## How to access parameters of module TSX STZ 10

### Introduction

This operation describes how to access the link configuration parameters for remote Nano PLCs via module TSX STZ 10 for TSX Micro PLCs.

### How to access the link

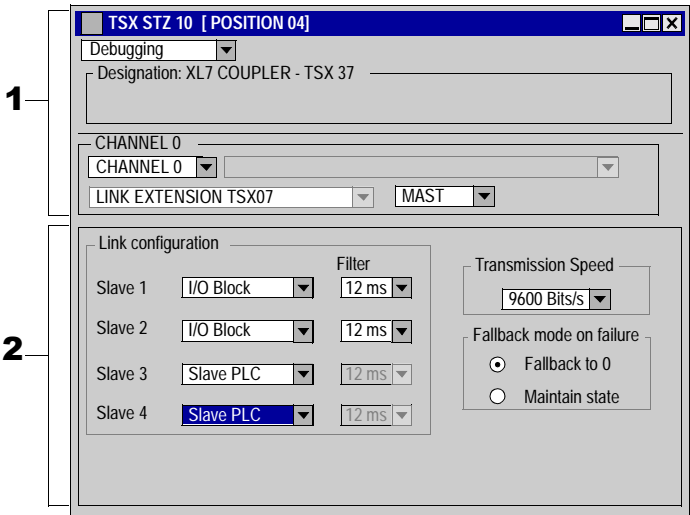
The following table shows the procedure for accessing the link:

Step	Action
1	<div>Declare module STZ 10 in slot 4</div> <div>Result</div> <div></div>
2	<div>Double click on the module to be configured, or select the module STZ 10 then the command <b>Services</b> → <b>Open module</b></div>

## Configuration screen for remoting Nano PLCs

**Introduction** This screen, split into two zones, is used to declare the communication channel and to configure the parameters necessary for a Modbus/Jbus link.

**Illustration** The screen for remoting Nano PLCs looks like this:



**Elements and functions**

This table describes the different zones that make up the configuration screen:

Address	Zone	Function
1	common	See Description of configuration screens for communication, p. 168.
2	specific	Is used to select or complete parameters for communication. It is split into three different types of information: <ul style="list-style-type: none"><li>● parameters concerning links,</li><li>● transmission speed,</li><li>● fallback mode on failure,</li></ul>

## Modbus parameters associated with the application

### Introduction

After having configured the communication channel, the link parameters must be completed.

They are split into three windows:

- the **Link configuration** window,
- the **Transmission speed** window,
- the **Fallback mode on failure** window.

### Link configuration

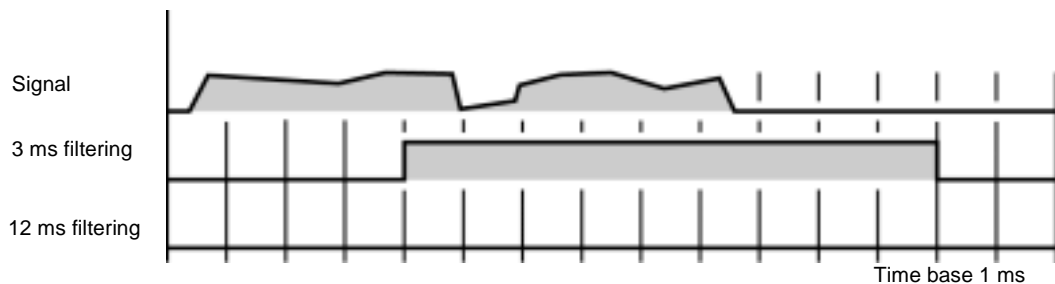
The window looks like this:

Link configuration		Filter
Slave 1	I/O Block	12 ms
Slave 2	I/O Block	12 ms
Slave 3	Slave PLC	12 ms
Slave 4	Slave PLC	12 ms

It can be used to select:

- the type of function chosen for the n slave selected:
  - Missing: there is no module on this link
  - I/O block: TSX Nano PLC is being used for input/output
  - Slave PLC: TSX Nano PLC is being used as a PLC
- filtering: associated with each I/O Block type slave, this function assigns 3 or 12 ms filtering to each input (default value) in order to avoid certain disruptions during acquisition of these inputs. The length of filtering time represents the minimum time that input must retain the same value in order to be taken into account.

Example





<b>Note:</b> While configuring the TSX 07 extension link, TSX AMN 4000 and TSX AMN 4001 analog modules must be declared as slave PLCs.
--

---

**Transmission speed**

Corresponds to the information transmission speed.

It must take on the value of 9600 or 19200 bits/s in the event that a link is made up only of PLC slaves or a mixed link (e.g. PLC slaves and I/O Block).

Its value can be raised to 38400 bits/s in the event that a link with only I/O Blocks.

---

**Fallback mode**

This parameter is used to configure the output fallback mode for all I/O blocks on the link (Fallback to 0 or state is maintained).

---



---

## Introduction

### Subject of this Chapter

This Chapter describes the Programming process during set-up of TSX Nano PLC remoting.

### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Example of communication with Nano PLCs	204
Configuring and programming the example	205

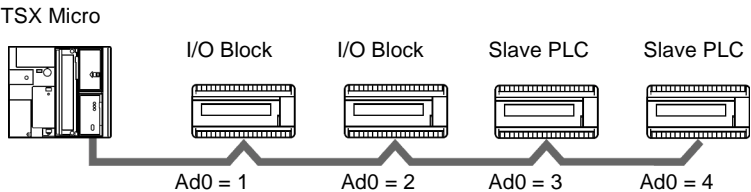
## Example of communication with Nano PLCs

### Introduction

The aim of this example is to demonstrate the use of two internal words of the TSX 37 (%MW0 and %MW1) as a shift register. Each of these words is transmitted to the TSX 07 Nano PLCs configured in slave mode. The program of these Nano PLCs consists solely of copying the input words onto the output words (%QW0.0:= %IW0.0.....).

### Illustration

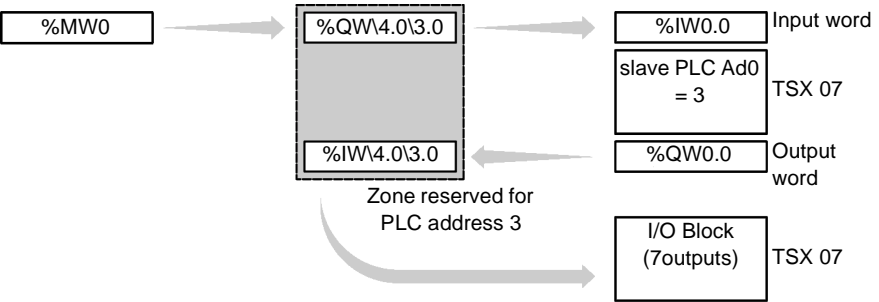
Configuration of the example is as follows:



### Synoptic of operation

The TSX 37 PLC therefore retrieves the output words of the two slave PLCs to transmit them onto the outputs of the two PLCs configured in I/O blocks. The TSX 37 makes a rotate shift of the words %MW0 and %MW1 every second.

The synoptic of operation for the word %MW0 is as follows:



## Configuring and programming the example

### Configuring the TSX STZ 10 module

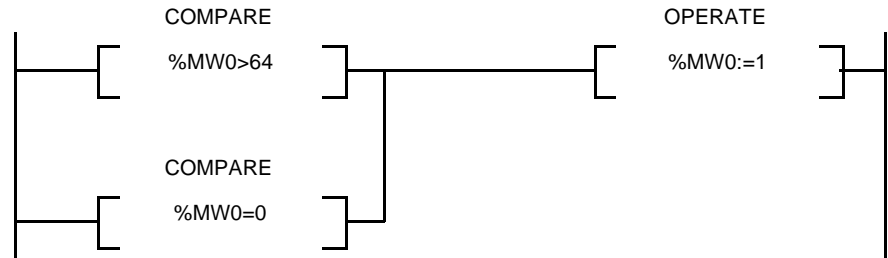
Configuration of the TSX STZ 10 module is as follows:

The screenshot shows the 'TSX STZ 10 [ POSITION 04]' configuration window. It includes a 'Configuration' dropdown, a 'Designation' field set to 'XL7 COUPLER - TSX 37', and a 'CHANNEL 0' section with a 'CHANNEL 0' dropdown and a 'LINK EXTENSION TSX07' dropdown. The 'Link configuration' section contains a table for Slave 1 through Slave 4, with columns for the slave type and a 'Filter' value of 12 ms. To the right of the table are settings for 'Transmission Speed' (9600 Bits/s) and 'Fallback mode on failure' (radio buttons for 'Fallback to 0' and 'Maintain state').

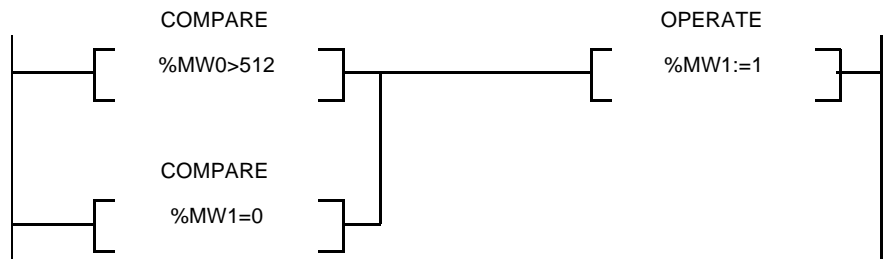
Slave	Configuration	Filter
Slave 1	I/O Block	12 ms
Slave 2	I/O Block	12 ms
Slave 3	Slave PLC	12 ms
Slave 4	Slave PLC	12 ms

## Programming

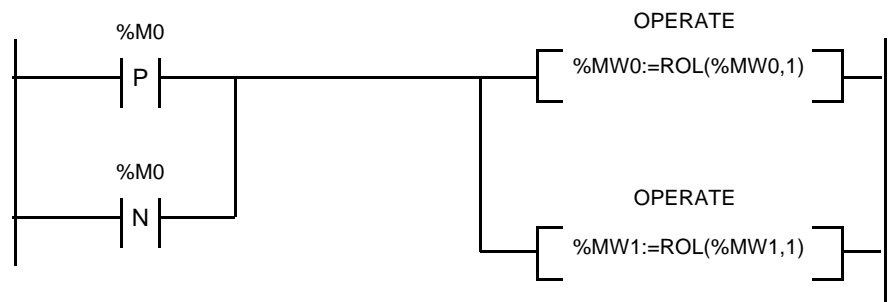
### Management of the change limits for %MW0



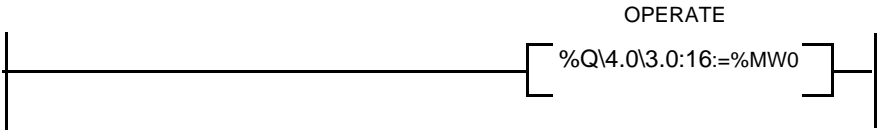
### Management of the change limits for %MW1



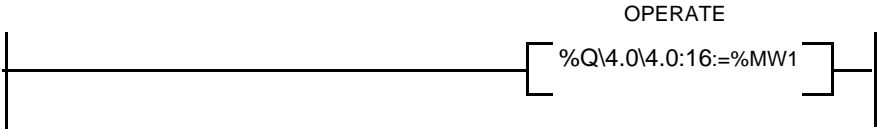
### Time delay of a second



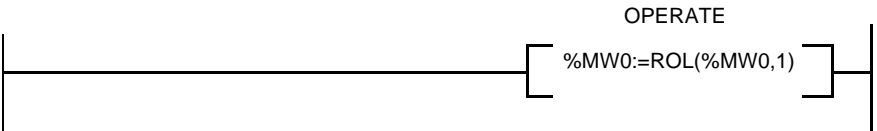
The rising or falling edge of %M0 generates the rotate shift of one bit to the left of %MW0 and %MW1.



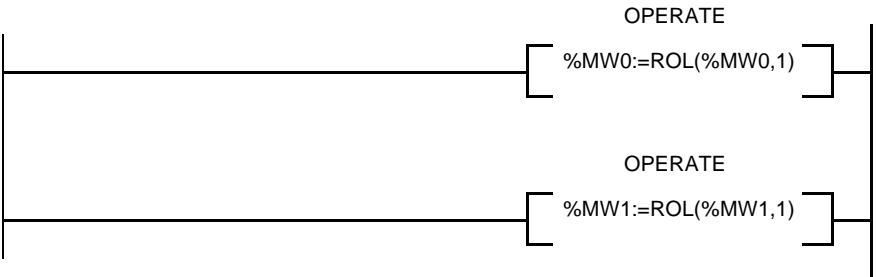
The internal word %MW0 is put in the output word %Q\4.0\3.0:16 assigned to the PLC at address 3. This output word will correspond to the input word %IW0.0.



The internal word %MW1 is put in the output word %Q\4.0\4.0:16 assigned to the PLC at address 4. This output word will correspond to the input word %IW0.0.



Copy to the input/output blocks







---

# Debugging Nano PLC remoting

## 9

---

### Introduction

#### Subject of this Chapter

This chapter describes the Debugging process during set-up of TSX Nano PLC remoting.

#### What's in this chapter?

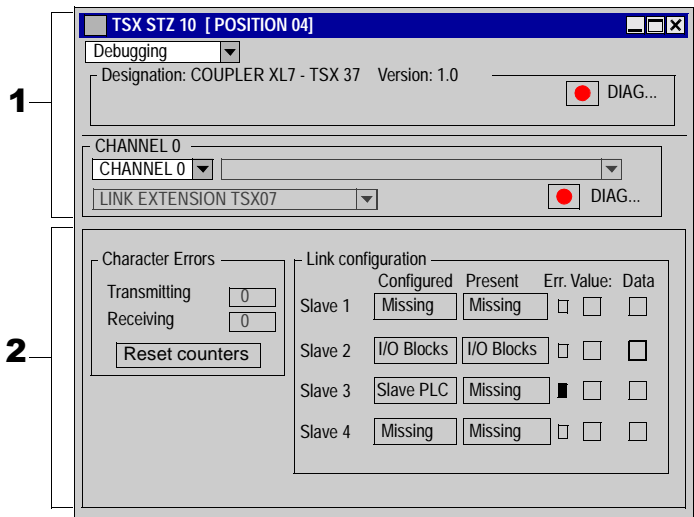
This chapter contains the following topics:

Topic	Page
Debugging screen for remoting Nano PLCs	210
Nano PLC Offset Debugging Screen	211

## Debugging screen for remoting Nano PLCs

**Introduction** This screen, split into two zones, is used to declare the communication channel and to configure the necessary parameters for communication with Nano PLCs.

**Illustration** The screen dedicated to communication looks like this:



**Elements and functions** This table describes the different zones that make up the debugging screen:

Address	Zone	Function
1	common	See Description of communication debugging screens, p. 170.
2	specific	Is used to access link debugging parameters.

---

## Nano PLC Offset Debugging Screen

---

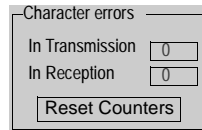
### At a Glance

This dedicated part is split into two windows:

- the **Character Errors** window,
  - the **Link Configuration** window.
- 

### Character Errors Window

This window looks like this:



This window shows the number of communication errors counted by the TSX STZ 10 module.

The **In Transmission** field corresponds to the number of transmission errors (%MW4.0.7 word image).

The **In Receive** field corresponds to the number of reception errors (%MW4.0.8 word image).

The **Zero Counters** button resets these counters to zero (bit %MW4.0.9:x0 is set to 1 and causes explicit exchange for updating).

**Note:** To carry out the same operations from the application, you must:

- activate the `READ_STS %CH4.0` function (updating words %MW4.0.7 and %MW4.0.8),
  - set bit %MW4.0.9:x0 to 1 and then execute the `WRITE_CMD %CH4.0` function.
- NB: setting bit %MW4.0.9:X0 to 1 does not set the default counters to zero (it does not start an explicit exchange).

## Link configuration window

This window looks like this:

	Configured	Present	Err.	Diag.	Data
Slave 1	Absent	Absent	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Slave 2	I/O Blocks	I/O Blocks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Slave 3	Slave AP	Absent	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Slave 4	Absent	Absent	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

This window allows you to test communication with the PLCs present on the link.

The different fields are:

- **Configured:** this shows the type of slave configured at the link address.
- **Present:** this shows whether the configured slave is physically present at the link address. The **Absent** value shows that the slave is not connected.
- **Err.:** if the slave is faulty, the LED is lit (in Reverse Video).
- **Diag.:** when a slave is faulty, this button allows you to access the diagnostics screen, showing the causes of the errors.
- **Data:** pushing this button displays the inputs/outputs image of the slave concerned.

**Note:** Where a slave has a physical address greater than 4, all the **Err.** LEDs are lit (Reverse Video).

## Data Example

Data example of an AP slave device:

3 - AP SLAVE Application Data	
%IW4.0/3.x	
MOT 0	16#2FCE
MOT 1	16#0240
%QW4.0/3.x	
MOT 0	16#0016
MOT 1	16#0000
Close	

Data example of an input/output Bloc device:

2 - I/O SLAVE Application Data	
%IW4.0/2.x	
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15	0 0 0 0 0 F1 0 0 0 0 0 0 0 0 0
%QW4.0/2.x	
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
F0= bit forced to 0 F1= bit forced to 1	
Close	

---

# Language objects associated to Nano PLC remoting

10

---

## Introduction

**Subject of this section**

This chapter introduces the language objects associated with the TSX Nano PLC remoting.

**What's in this chapter?**

This chapter contains the following topics:

Topic	Page
Language object for implicit exchange	214
Explicit exchange language objects	215
Explicit exchange management and reports	218
Language objects associated with configuration	219

## Language object for implicit exchange

---

### Introduction

This page describes all the language objects in implicit exchange for the remoting of TSX Nano PLCs that can be displayed or modified by the application program.  
Default exchanges, p. 155

---

### Bit objects

The table below shows the different bit objects for implicit exchange.

Object (1)	Function	Meaning
%I4.MOD.ERR	Module error bit	This bit set to 1, indicates a module error (at least one of the channels is faulty,...)
%I4.0.ERR	Status of link	= 1: if there is a configuration error or faulty devices
%I4.0i.n %Q4.0i.n	Values of remote inputs/ outputs	Value of the inputs Value of the outputs
Key		
(1)	Address i.n <ul style="list-style-type: none"><li>● i: corresponds to the slave number</li><li>● n: corresponds to the word number</li></ul>	

---

### Word objects

The table below shows the different word objects for implicit exchange.

Object (1)	Function	Meaning
%IW4.0.0	Device status	<ul style="list-style-type: none"><li>● x0 = 1: if configuration error</li><li>● xn = 1: if the device is faulty (if = 0 if not)</li><li>● x5 = 1: if at least one of the devices is faulty</li></ul>
%IW4.0i.n %QW4.0i.n	Value of the application data	Words consumed by master Words produced by master
Key		
(1)	Address i.n <ul style="list-style-type: none"><li>● i: corresponds to the slave number</li><li>● n: corresponds to the word number</li></ul>	

---

## Explicit exchange language objects

### Introduction

This page describes all the language objects in explicit exchange for reporting Nano TSX PLCs that can be displayed or modified by the application program. Specified exchanges : General points, p. 157

### Word objects

The table below shows the different word objects for explicit exchange.

Object (1)	Function	Meaning
%MW4.MO D.2	Module status	<ul style="list-style-type: none"><li>● x0 = 1: defective module</li><li>● x1 = 1: functional error (error between the processor and the module, adjustment or configuration error, ...)</li><li>● x2 = 1: terminal block fault (not connected)</li><li>● x3 = 1: self-tests running</li><li>● x4 = 1: reserved</li><li>● x5 = 1: error in hardware or software configuration (the module present is not that declared in the configuration, the sub-modules are not compatible)</li><li>● x6 = 1: missing module</li><li>● x7 = 1: error in one of the sub-modules</li></ul>
%MW4.0.2	Standard channel status	<ul style="list-style-type: none"><li>● x0 = 1: single station on the network</li><li>● x1 = 1: not used</li><li>● x2 = 1: transmission line error</li><li>● x3 = 1: not used</li><li>● x4 = 1: internal software fault</li><li>● x5 = 1: error in hard- or software configuration, or configuration missing</li><li>● x6 = 1: communication error with the processor</li><li>● x7 = 1: application fault (double address station)</li></ul>

Object (1)	Function	Meaning
%MW4.0.3 %MW4.0.4	Specific channel status	<p>One byte per device (byte 0 belongs with device 1, ..., byte 3 with device 4) If an octet is worth</p> <ul style="list-style-type: none"> <li>● 16#00: correct exchange</li> <li>● 16#02: character send error</li> <li>● 16#03: character receive error</li> <li>● 16#04: Protocol Data Unit ERROR</li> <li>● 16#05: type inconsistency (I/O and PLC)</li> <li>● 16#06: Protocol Data Unit incorrect</li> <li>● 16#07: Binary Check Character error</li> <li>● 16#08: timeout error</li> <li>● 16#09: time between characters error</li> <li>● 16#81: protected outputs fault</li> <li>● 16#82: supply fault</li> </ul>
%MW4.0.5	Physical configuration of slaves 1 and 2	<p>Byte 0: Slave 1 value Byte 1: Slave 2 value For each byte, bits x0 and x1 give:</p> <ul style="list-style-type: none"> <li>● x10 = 00: slave missing</li> <li>● x10 = 01: PLC slave</li> <li>● x10 = 10: input/output slave</li> </ul> <p>For each byte, bits x2 and x3 correspond to the input/output block:</p> <ul style="list-style-type: none"> <li>● x3x2 = 00: 6 inputs, 4 outputs</li> <li>● x3x2 = 01: 9 inputs, 7 outputs</li> <li>● x3x2 = 10: 14 inputs, 10 outputs</li> <li>● x3x2 = 11: alternating, 9 inputs, 7 outputs</li> </ul>
%MW4.0.6	Physical configuration of slaves 3 and 4	<p>Byte 0: Slave 3 value Byte 1: Slave 4 value For each byte, bits x0 and x1 give:</p> <ul style="list-style-type: none"> <li>● x10 = 00: slave missing</li> <li>● x10 = 01: PLC slave</li> <li>● x10 = 10: input/output slave</li> </ul> <p>For each byte, bits x2 and x3 correspond to the input/output block:</p> <ul style="list-style-type: none"> <li>● x3x2 = 00: 6 inputs, 4 outputs</li> <li>● x3x2 = 01: 9 inputs, 7 outputs</li> <li>● x3x2 = 10: 14 inputs, 10 outputs</li> <li>● x3x2 = 11: alternating, 9 inputs, 7 outputs</li> </ul>
%MW4.0.7	Error counter	Send error
%MW4.0.8	Error counter	Receive error



Object (1)	Function	Meaning
%MW4.0.9	Commands	<ul style="list-style-type: none"><li>● x0 = 1: Reset counters</li><li>● x1 = 1: reactivation of slave 1 outputs</li><li>● x2 = 1: reactivation of slave 2 outputs</li><li>● x3 = 1: reactivation of slave 3 outputs</li><li>● x4 = 1: reactivation of slave 4 outputs</li></ul>

---

## Explicit exchange management and reports

---

### Introduction

This page describes all the language objects that manage explicit exchanges. Exchange and report management, p. 159

---

### Word objects

The table below shows the different word objects for the management of explicit exchanges.

Object	Function	Meaning
%MW4.MO D.0	Module exchanges in progress	<ul style="list-style-type: none"><li>● x0 = 1: reading status in progress</li><li>● x1 = 1: sending of command parameters to the communication module</li><li>● x2 = 1: sending of adjustment parameters to the communication module</li></ul>
%MW4.MO D.1	Module report	<ul style="list-style-type: none"><li>● x1 = 0: command parameters received and accepted by the module</li><li>● x2 = 0: adjustment parameters received and accepted by the module</li></ul>
%MW4.0.0	Channel exchanges in progress	<ul style="list-style-type: none"><li>● x0 = 1: reading status in progress</li><li>● x1 = 1: sending of command parameters to the communication channel</li><li>● x2 = 1: sending of adjustment parameters to the communication channel</li></ul>
%MW4.0.1	Channel report	<ul style="list-style-type: none"><li>● x1 = 0: command parameters received and accepted by the communication channel</li><li>● x2 = 0: adjustment parameters received and accepted by the communication channel</li></ul>

---

## Language objects associated with configuration

### Introduction

This page describes all the configuration language objects for remoting Nano TSX PLCs that can be displayed by the application program.

### Internal constants

The following table describes the internal constants:

Object	Function	Meaning
%KW4.0.1	Speed / Format	<ul style="list-style-type: none"> <li>● = 16#03: if speed is 9600 bit/s</li> <li>● = 16#04: if speed is 19200 bit/s</li> <li>● = 16#05: if speed is 38400 bit/s</li> </ul>
%KW4.0.2 %KW4.0.3	Logic configuration	One byte per device (byte 0 corresponding to device 1, ..., byte 3 to device 4) <ul style="list-style-type: none"> <li>● = 16#00: Missing</li> <li>● = 16#01: slave PLC</li> <li>● = 16#00: Input/output block</li> </ul>
%KW4.0.4	Filter Fallback	Byte 0: corresponds to filtering (2 bits per device) <ul style="list-style-type: none"> <li>● = 01: 3 ms filter</li> <li>● = 11: 12 ms filter</li> </ul> Byte 1: corresponds to fallback mode (2 bits per device) <ul style="list-style-type: none"> <li>● x0 = 0: outputs fallback to 0</li> <li>● x0 = 1: outputs maintained</li> </ul>



---

# Communication using character mode



---

## Introduction

### Subject of this part

This part introduces the principles of configuring and using communication by character mode via PL7 software.

### What's in this part?

This part contains the following chapters:

Chapter	Chaptername	Page
11	General	223
12	Configuring character mode communication	233
13	Programming communication in character mode	247
14	Debugging communication using character mode	249
15	Language objects associated with character mode communication	255



Introduction

Subject of Chapter

This Chapter introduces communication via character mode and its services.

What's in this chapter?

This chapter contains the following Sections:

Section	Topic	Page
11.1	Introduction to communication via character mode	224
11.2	Characteristics	228

11.1

Introduction to communication via character mode

---

Introduction

---

Subject of Section

This Section provides a summary description of communication via character mode and its associated services.

---

What's in this section?

This section contains the following topics:

Topic	Page
About character mode	225
Flow control	226

---



## About character mode

---

### Introduction

Communication via character mode is used to perform dialog and communication functions between PLCs and their environment:

- standard peripherals: printers, screen with keypad, workshop terminal,
  - specialized peripherals: barcode scanners,
  - link to a production control or management computer,
  - data transmission between mixed devices (digital commands, speed controllers etc.),
  - link to an external modem.
- 

### Associated manuals

If you require further information you should consult the following manuals:

Title	Description
TSX Micro PLCs - Installation manual	Hardware installation
Premium TSX PLCs - Installation manual	Hardware installation

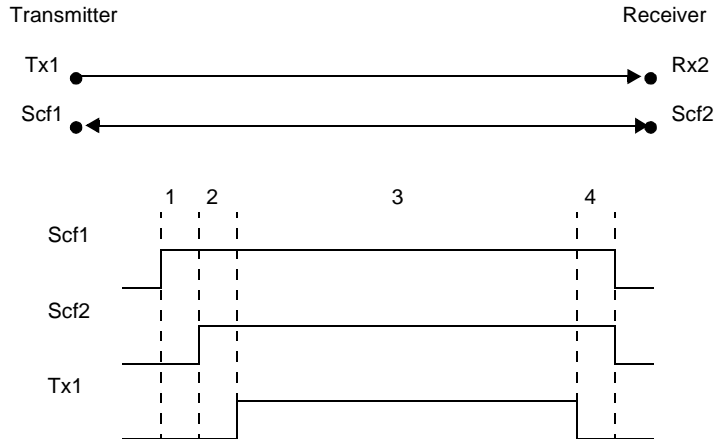
---

## Flow control

### Introduction

Flow control is used to manage exchanges on a serial link (in this case a character mode link) between two devices.

Data is sent from the transmitter Tx1 to the receiver Rx2. Data transmission is controlled by the flow control signals Scf1 and Scf2.



- 1 The transmitter activates its Scf1 signal to show that it is ready to send.
- 2 The receiver activates its Scf2 signal to allow data to be sent.
- 3 Data is sent
- 4 Once data transmission has finished, Scf1 and Scf2 control signals are deactivated.

There are two ways to control flow:

- either with hardware using:
  - RTS/CTS
  - RTS/DCD
- or with software via Xon/Xoff.

**Note:** Flow is most commonly controlled using software. Should software control not be available, the operation will be carried out using hardware.

**RTS/CTS**

In this case, the control signals are RTS/CTS signals. This flow control mode is the most commonly used type of hardware flow control.

The output Tx of the transmitter is linked to the Rx input of the receiver, and vice versa. The CTS signal from the transmitter is linked to the RTS receiver signal, and vice versa.

The transmitter is allowed to send data when it receives the RTS signal from the transmitter on its CTS input.

---

**RTS/DCD**

In this case, the control signals are RTS/DCD signals. This flow control mode is not widely used. It is there for communication with, for example, a bottom-of-the-range printer.

The output Tx of the transmitter is linked to the Rx input of the receiver, and vice versa. The DCD signal from the transmitter is linked to the DTR signal from the receiver. The RTS signal from the transmitter is linked to the CTS signal from the receiver.

The transmitter is allowed to send data when it receives the RTS signal from the receiver on its CTS input.

---

**Xon/Xoff**

In this case, this flow control is carried out by software with Xon/Xoff characters. In this instance, devices are linked by two lines only.

The output Tx of the transmitter is linked to the Rx input of the receiver, and vice versa.

The transmitter is allowed to send data when it receives the Xon character on its Rx input. When its Rx input receives the Xoff character, it must halt transmission.

---

# 11.2                    Characteristics

---

## Introduction

**Subject of Section**                    This Section introduces the characteristics of using communication via character mode.

---

**What's in this section?**                    This section contains the following topics:

Topic	Page
Compatibility	229
Performance	230
Operating mode	232

---

## Compatibility

---

### Hardware

This type of communication is available for:

- TSX Micro and TSX Premium PLCs via the terminal port associated with the RS485 physical layer,
  - TSX Premium via:
    - the PCMCIA TSX SCP11 card associated with the RS232 physical layer,
    - the PCMCIA TSX SCP 112 card associated with 20 mA current loops,
    - the PCMCIA TSX SCP 114 card associated with RS422 and RS485 physical layers,
    - the built-in link on the TSX SCY 21600 / 21601 module associated with the physical layer RS485,
  - TSX Micro accepts the PCMCIA cards described below.
- 

### Software

The terminal port of TSX Premium and TSX Micro processors can only process one communication function type:

- INPUT\_CHAR,
- PRINT\_CHAR,
- OUT\_IN\_CHAR.

For each terminal port communication, the maximum frame size is 120 bytes per communication function.

PCMCIA cards can process:

- 4 communication functions simultaneously for TSX Micro PLCs,
- 8 communication functions simultaneously for TSX Premium PLCs.

The built-in link on TSX SCY 21600/21601 modules can process 8 communication functions at the same time.

For each PCMCIA card or built-in link communication, the maximum frame size is 4Kbytes per communication function.

---

# Performance

## Introduction

The following tables are used to assess the typical exchange time in character mode for:

- PCMCIA cards and the built-in link on TSX SCY 21600/21601 modules,
- the terminal port.

The results shown correspond to the average time (in ms) for the PRINT\_CHAR function to be executed.

## Time with PCMCIA card

Average time according to programmed cycle time and number of characters sent:

Length of message		80 characters		960 characters	
Speed in bits/s	T cycle in ms	Average time		Average time	
		PCMCIA	SCY 21600	PCMCIA	SCY 21600
4800	10	190	210	2100	2200
4800	25	200	220	2166	2300
4800	50	200	230	2300	2400
9600	10	108	125	1120	1200
9600	25	118	135	1147	1230
9600	50	137	157	1148	1240
19200	10	62	90	604	700
19200	25	75	105	696	800
19200	50	100	120	698	810

**Time with  
terminal port**

Average time according to programmed cycle time and with 80 characters sent:

		<b>TSX 37</b>	<b>TSX 57</b>
Speed in bits/s	T cycle in ms	Average time	Average time
1200	10	939	939
1200	20	945	945
1200	50	948	948
1200	100	1000	1000
1200	255	1018	1018
4800	10	242	242
4800	20	242	242
4800	50	249	249
4800	100	299	299
4800	255	455	455
9600	10	129	129
9600	20	139	139
9600	50	149	149
9600	100	199	199
9600	255	355	355
19200	10	-	65
19200	20	-	75
19200	50	-	105
19200	100	-	155
19200	255	-	285

## Operating mode

---

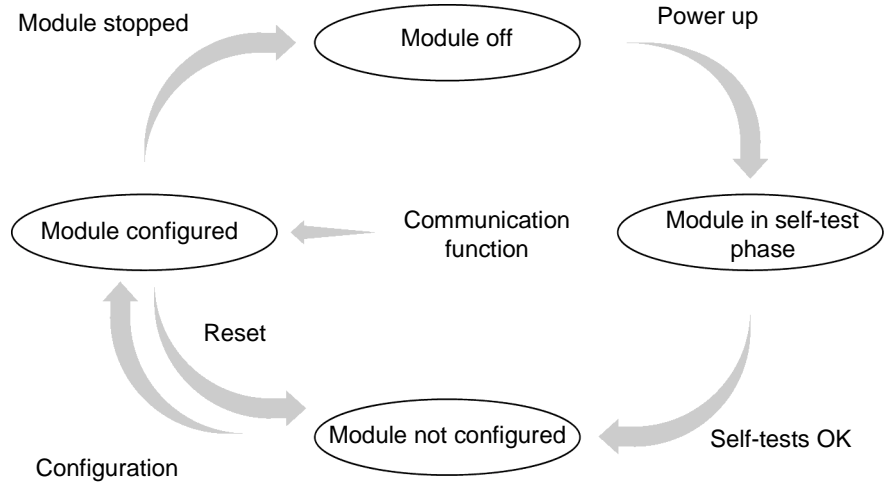
### Introduction

The following graph describes the operating modes in character mode for PCMCIA cards, for the built-in link on TSX SCY 21600/21601 modules and the terminal port.

---

### General graph

The operating mode is as follows:



### Operation

- after power-up the module self-tests. During this phase the warning indicators flash.
  - if there is no PL7 application in the PLC, the module remains in await configuration mode.
  - if there is a PL7 application in the PLC, the application configuration is sent to the module, which then starts.
  - in case of power outage, the PLC processor performs a warm restart. The module then re-launches its self-test procedure.
-



---

# Configuring character mode communication

12

---

## Introduction

### Subject of this Chapter

This Chapter describes the Configuration process during set-up of character mode communication.

### What's in this chapter?

This chapter contains the following topics:

Topic	Page
How to access PCMCIA card parameters in character mode	234
How to access terminal port parameters	235
How to access TSX SCY 21600 / 21601 module parameters	236
Configuration screen in character mode	237
Functions accessible in character mode	238
Parameters in character mode linked to transmission	239
Parameters in character mode linked to message ends	241
Parameters in character mode linked to flow control	243
Additional parameters	244

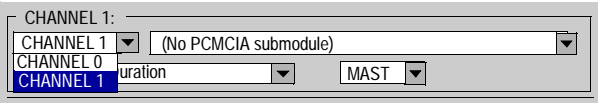
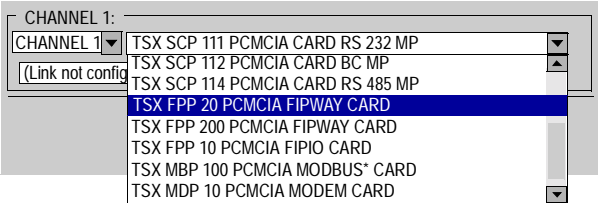
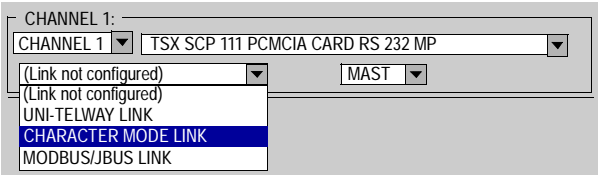
## How to access PCMCIA card parameters in character mode

### Introduction

This operation describes how to access configuration parameters for the character mode link via PCMCIA cards.

### How to access the link

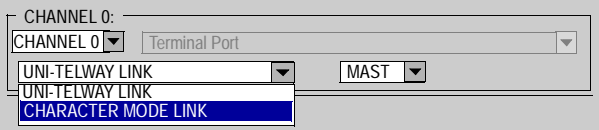
The following table shows the procedure for accessing the character mode link:

Step	Action
1	Access the communication channel configuration screen.
2	Select the communication channel from the drop-down menu <b>CHANNEL 1</b> <b>Example</b> 
3	Select one of the following PCMCIA cards from the drop-down menu: <ul style="list-style-type: none"><li>● <b>TSX SCP 111 PCMCIA CARD RS232 MP</b></li><li>● <b>TSX SCP 112 PCMCIA CARD BC MP</b></li><li>● <b>TSX SCP 114 PCMCIA CARD RS485 MP</b></li></ul> <b>Example</b> 
4	Select the link from the drop-down menu <b>CHARACTER MODE LINK:</b> <b>Example</b> 

# How to access terminal port parameters

**Introduction** This operation describes how to access configuration parameters for the character mode link via the terminal port.

**How to access the link** The following table shows the procedure for accessing the character mode link:

Step	Action
1	Access the communication channel configuration screen.
2	Select the link from the drop-down menu <b>CHARACTER MODE LINK</b> . <b>Example</b> 

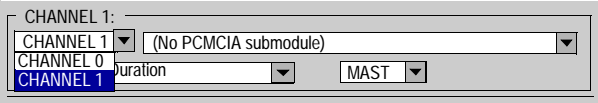
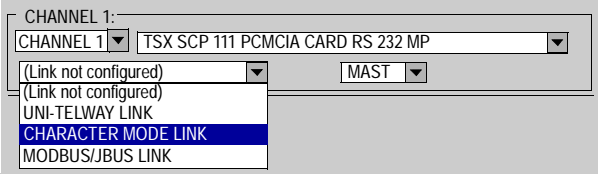
## How to access TSX SCY 21600 / 21601 module parameters

### Introduction

This operation describes how to access configuration parameters for the character mode link via TSX SCY 21600 / 21601 modules for TSX Premium.

### How to access the link

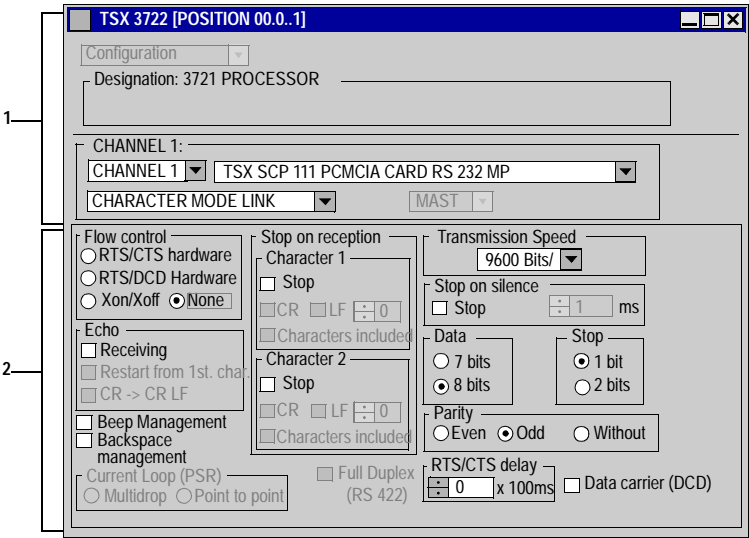
The following table shows the procedure for accessing the character mode link:

Step	Action
1	Access the communication channel configuration screen of the chosen module
2	Select the communication channel from the drop-down menu <b>CHANNEL 0</b> <b>Example</b> 
3	Select the link from the drop-down menu <b>CHARACTER MODE LINK</b> . <b>Example</b> 

# Configuration screen in character mode

**Introduction** This screen, split into two zones, is used to declare the communication channel and to configure the parameters necessary for a character mode link.

**Illustration** The screen dedicated to character mode communication looks like this:



## Elements and functions

This table describes the different zones that make up the configuration screen:

Address	Zone	Function
1	common	See Description of configuration screens for communication, p. 168.
2	specific	Is used to select or complete the parameters of a character mode link. It is split into four different types of information: <ul style="list-style-type: none"><li>the transmission parameters.</li><li>parameters concerning end of message detection,</li><li>parameters concerning flow control,</li><li>additional parameters.</li></ul>

## Functions accessible in character mode

### Introduction

Depending on the chosen communication supports, some parameters cannot be modified. They appear grayed out.

### Accessible functions

The summary table below shows the various choices possible:

Functions	SCP 111	SCP 112	SCP 114	SCY 21600/ 21601	Terminal Port
Flow control	<ul style="list-style-type: none"> <li>● RTS/CTS</li> <li>● RTS/DCD</li> <li>● Xon/Xoff</li> <li>● None</li> </ul>	No	No	No	No
Echo	<ul style="list-style-type: none"> <li>● On reception</li> <li>● Restart from 1st. char.</li> <li>● CR-&gt;CRLF</li> </ul>	<ul style="list-style-type: none"> <li>● On reception</li> <li>● Restart from 1st. char.</li> <li>● CR-&gt;CRLF</li> </ul>	No	No	On reception
Current Loop (PSR)	No	Yes	No	No	No
Stop on reception	Yes	Yes	Yes	Yes	<ul style="list-style-type: none"> <li>● CR/LF with 1 Micro</li> <li>● No with 1 Premium</li> </ul>
Full duplex	No	No	Yes	No	No
Transmission speed	Yes	Yes	Yes	Yes	Yes
Stop on silence	Yes	Yes	Yes	Yes	No
Data / Stop	Yes	Yes	Yes	Yes	Yes
Parity	Yes	Yes	Yes	Yes	Yes
RTS/CTS delay Data carrier (DCD)	Yes	No	No	No	No

**Beep** and **Back-space** management can be accessed whatever the type of support.

---

## Parameters in character mode linked to transmission

---

### Introduction

Once the communication channel has been configured, complete the parameters dedicated to transmission.

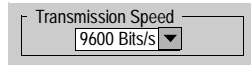
They are split into four windows:

- the **Transmission speed** window,
- the windows specific to **Data** and **Stop**,
- the **Parity** window,
- the **RTS/CTS delay** window,

---

### Transmission speed

The window looks like this:



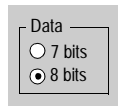
It is used to select the transmission speed for the character mode protocol used by the module:

- the default speed is 9600 bits/s.
- the available speeds are 1200, 2400, 9600 and 19200 bits/s.
- speeds of 300 and 600 bits/s are only available with a PCMCIA TSX SCP 111 card.
- you are advised to adjust transmission speed according to the remote device.

---

### Data

The window looks like this:



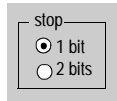
The **Data** fields specify the size of data that can be exchanged on the line. 7 and 8 bit values are available. You are advised to adjust the number of data bits according to the remote devices.

**Note:** The default value is 8 bits.

---

**Stop**

The window looks like this:

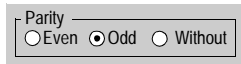


The **Stop** field is used to complete the number of stop bits employed for communicating in character mode. Possible values are 1 or 2 stop bits. You are advised to adjust the number of stop bits according to the remote devices.

**Note:** The default value is 1 stop bit.

**Parity**

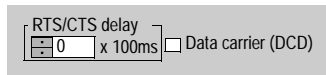
The window looks like this:



This field is used to define the addition or non-addition of a parity bit, and its type. Possible values are Even, Odd, or none (Odd by default). It is advisable to adjust the parity according to the remote stations.

**RTS/CTS delay**

The window looks like this:



Before each transmission of a character string, the module activates the RTS signal (Request To Send) and awaits the activation of the CTS signal (Clear To Send). It is used to complete:

- the maximum waiting time between the two signals. If this time has elapsed, the request is not sent onto the bus.
  - the value is expressed in hundreds of milliseconds,
  - The default value is 0 ms.
  - the value is between 0 and 10 s,
  - the value 0 specifies the absence of delay management between the two signals.
- the carrier management (DCD signal, Data Carrier Detected) is only used in the event of communication with a modem with a controlled carrier:
  - If the option is selected, character reception is only enabled if the DCD carrier signal is detected,
  - If the option is not selected, all the characters received are taken into account.



## Parameters in character mode linked to message ends

### Introduction

After having configured the communication channel, the end of message detection parameters must be completed.

They are split into two windows:

- the **Stop on reception** window: a stop condition triggered by a special character,
- the **Stop on silence** window: a stop condition triggered by a silence.

### Usage conditions

Activating one of these conditions leads to the following events:

- the **INPUT\_CHAR** communication function is not able to read of a defined number of characters. The **Number of characters to be read** parameter must be equal to 0.

- the **OUT\_IN\_CHAR** communication function may be used on reception.

Selection of stop on silence results in deselection of stop by character. Inversely, selection of stop by character will deselect stop on silence.

### Stop on reception

The window looks like this:

A receive request can be discontinued on receiving a particular character.

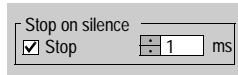
The following parameters are used to define up to two end of message characters:

- **Stop**: is used to activate stop on reception by an end character,
- **CR**: is used to detect the end of a message by a carriage return,
- **LF**: is used to detect the end of a message by a line feed,
- an entry field: is used to identify an end character which is different from a CR or LF character, using a decimal value.

The possible values are:

- 0 to 255 if the data is coded onto 8 bits,
- 0 to 127 if the data is coded onto 7 bits,
- **Character included**: check the box if you wish the end character(s) to be included in the receive table for the PL7 PLC application.

**Stop on silence**      The window looks like this:



The image shows a small configuration window titled "Stop on silence". Inside the window, there is a checkbox labeled "Stop" which is checked. To the right of the checkbox is a numeric input field containing the value "1", followed by the unit "ms".

This parameter is used to detect the end of a message on reception by the absence of an end character for a given time period.  
Stop on silence is enabled when the **Stop** box is checked. The duration of the silence (in milliseconds) is set in the entry field.

**Note:** Available values are between 1 ms and 10000ms.

## Parameters in character mode linked to flow control

---

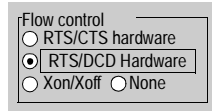
### Introduction

Once the communication channel has been configured, the flow control parameters must be entered. See Flow control, p. 226.

---

### Flow control window

The window looks like this:



Selection of flow control depends on the remote device:

- **RTS/CTS hardware:** if the device manages this flow control.
  - **RTS/DCD hardware:** if the device manages this flow control.
  - **Xon/Xoff:** if the device manages this flow control.
  - **None:** if the device does not manage flow control.
-

## Additional parameters

---

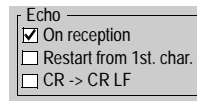
### Introduction

Configuring a link using character mode requires the following four parameters to be configured:

- the **Echo** window,
- the **Beep management** parameter,
- the **Back-space management** parameter,
- the **Full Duplex (RS 422)** parameter,

### Echo

This window is used to select and configure reception echo management.



Any character received by the PLC is immediately resent as an echo on the line (which allows the remote device to carry out a control procedure).

To enable echo management, check the **On reception** box.

If, during reception, a write request is sent by the PLC, echo on reception is interrupted. When the write request has finished, the echo is restarted in two distinct ways:

- either from the first character received, in which case check the **Restart from 1st char. box**,
- or from the last interrupted character , in which case uncheck the **Restart from 1st char. box**,

Selecting the **CR --> CR LF** box is used to send as part of an echo the carriage return character followed automatically by the line feed character (LF = 16#0A) after reception of any carriage return character (CR = 16#0D).

### Beep management

Checking the **Beep management** box causes a beep to be heard when the module receive buffer is empty or full.



Uncheck the box if the card is connected to a MMI terminal.

**Backspace management**

Checking the **Backspace management** box ensures that each backspace character received is not stored and that the previous character is canceled. In addition, if the echo on reception is active, the PLC transmits three characters in the following order:

- Backspace (= 16#08)
- space (= 16#20)
- Backspace (= 16#08)

If the box is deselected, any Backspace character received is stored like any other character.

---

**Full Duplex (RS 422)**

Checking this box enables Full Duplex communication. Otherwise communication is Half Duplex. Activating this function depends on the remote device.

☐ Full Duplex  
(RS 422)

---



---

# Programming communication in character mode

# 13

---

## Available communication functions

### Introduction

This page describes the communication functions available in character mode, and presents an example of communication between two stations (TSX Micro and TSX Premium).

### Available functions

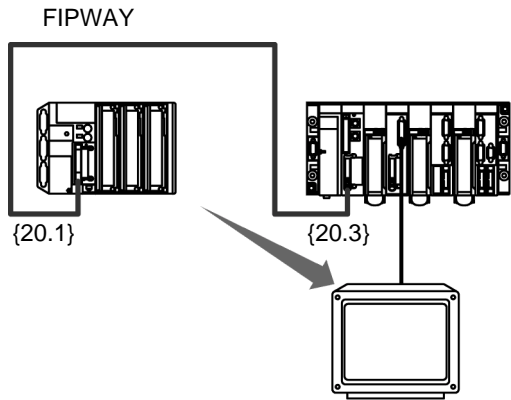
Three specific communication functions are defined to send and receive data on a communication channel in character mode:

- `PRINT_CHAR`: sends a character string. See Writing a string of characters. `PRINT_CHAR`, p. 117.
- `INPUT_CHAR`: character string read request. See Reading a character string: `INPUT_CHAR`, p. 122.
- `OUT_IN_CHAR`: sends a character string followed by a read request. See Sending/receiving a string of characters : `OUT_IN_CHAR`, p. 126.

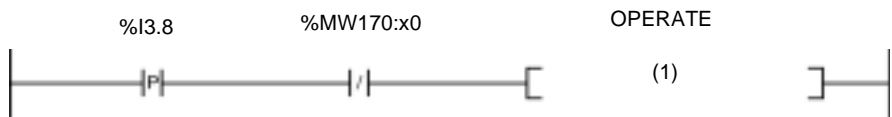
<b>Note:</b> Usage of these functions must be consistent with the configuration.
--

**Example**

A station at the address {20.1} on a FIPWAY network wants to send a character string to then receive a character string from a video terminal connected via the built-in link of a TSX SCY 21601 module at station address {20.3}.



Programming the communication function:



(1) OUT\_IN\_CHAR(ADR#{20.3}0.0.SYS, 1, %MB300:10, %MB310:10, %MW170:4)

The following table describes the different parameters of the function:

Parameter	Description
ADR#{20.3}0.0.SYS	Address of destination device for the message
1	Send / Receive
%MB300:10	Contents of message to be sent
%MB310:10	Contents of the message received
%MW170:4	Exchange report, length of string sent, then length of string received

**Note:** Before each function starts, the number of characters to be sent must be given in the length parameter (in bytes). In this example: %MW173 = 10. At the end of the exchange, it will contain the number of characters received (in bytes). The value 0 is used to send the entire character string.



---

# Debugging communication using character mode

14

---

## Introduction

### Subject of this Chapter

This Chapter describes the Debugging process during set-up of character mode communication.

### What's in this chapter?

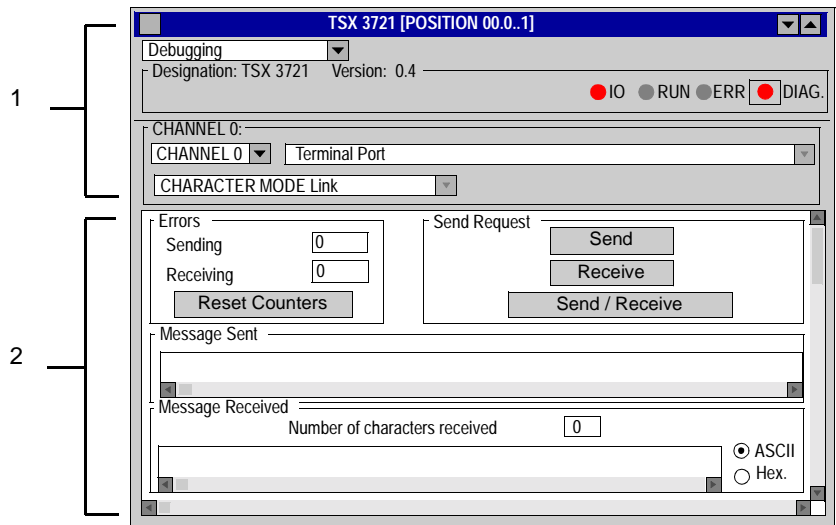
This chapter contains the following topics:

Topic	Page
Debugging screen in character mode	250
Debugging parameters in character mode	251
How to test a communication channel	253

## Debugging screen in character mode

**Introduction** This screen, split into two zones, is used to declare the communication channel and to configure the parameters required for a character mode link.

**Illustration** The screen dedicated to character mode communication looks like this:



**Elements and functions**

This table describes the different zones that make up the debugging screen:

Address	Zone	Function
1	common	Description of communication debugging screens, p. 170
2	specific	Is used to access the debugging parameters of a character mode link.

## Debugging parameters in character mode

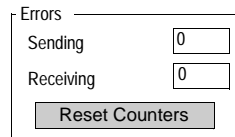
### Introduction

The specific part is split into four windows:

- the **Errors** window,
- the **Request transmission** window,
- the **Message sent** window,
- the **Message received** window,

### Errors Window

The window looks like this:



The Errors window is a rectangular box with a title bar labeled "Errors". Inside, there are two rows: "Sending" and "Receiving". Each row has a text label on the left and a small rectangular input field on the right, both containing the number "0". Below these two rows is a single button labeled "Reset Counters".

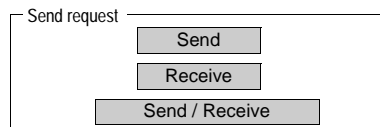
This window indicates the number of communication errors counted by the communication module.

- **On transmission:** corresponds to the number of errors on transmission (image of %MWxy.i.4 word)
- **On reception:** corresponds to the number of errors on reception (image of %MWxy.i.5 word)

The button **Reset Counters** resets these counters to zero.

### Request Transmission window

The window looks like this:



The Request Transmission window is a rectangular box with a title bar labeled "Send request". Inside, there are three buttons stacked vertically. The top button is labeled "Send", the middle button is labeled "Receive", and the bottom button is labeled "Send / Receive".

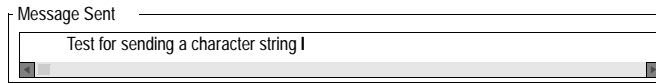
This window is used to test a communication channel by transmission and/or reception of a character string.

- the **Send** button transmits a character string.
- the **Receive** button is used to receive a character string.
- the **Send/Receive** button is used to send a character string and wait for a reply.

**Note:** Reception can be stopped by pressing the Escape button, or if a message is received.

### Message Sent window

The window looks like this:

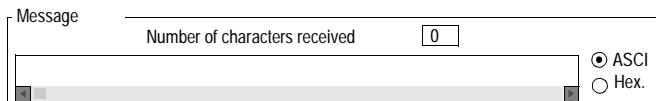


This window is used to enter a message to be sent during a communication test using the **Send** and **Send/Receive** buttons.

---

### Message Received window

The window looks like this:



This window is used to read a received message as a result of a communication test by using the **Receive** and **Send/Receive** buttons.  
The **ASCII** and **Hexa** buttons are used to display text in hexadecimal or in ASCII format.

---

---

## How to test a communication channel

---

### Introduction

This page describes the procedure for testing a communication channel from the debugging screen.

---

### How to send a character string

The following procedure is used to send a character string with a remote device.

Step	Actions
1	Enter the character string to be sent in the <b>Message sent</b> window.  <b>Note:</b> Special characters can also be sent. They must begin with the \$ character (example using carriage return character: \$0D).
2	Click on the <b>Send</b> button.  <b>Result</b> If the exchange is correct, a window specifying that the exchange is correct appears. On the remote device display, check whether the string has been transmitted.

---

### How to receive a character string

The following procedure is used to receive a character string with a remote device. For efficient operation, you must remember that this test requires stop on reception to be configured either via a special character, or via a silence.

Step	Action
1	Click on the <b>Receive</b> button.
2	Send the character string with the frame end character from the remote device.  <b>Note:</b> In this example, stop on reception is performed after a carriage return character (16#0D).
3	Display the number of characters and the character string received in the <b>Message received</b> window.

---



---

# Language objects associated with character mode communication

15

---

## Introduction

### Subject of this Section

This Chapter introduces language objects associated with character mode communication.

### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Language objects in implicit exchange	256
Explicit exchange language objects	257
Explicit exchange management and reports	259
Language objects associated with configuration	260

## Language objects in implicit exchange

---

### Introduction

This page describes all the language objects for implicit exchange in character mode communication that can be displayed or modified by the application program. See Default exchanges, p. 155.

---

### Bit objects

The table below shows the different bit objects for implicit exchange.

Object (1)	Function	Meaning
%lxy.MOD.ERR	Module error bit	This bit set to 1, indicates a module error (at least one of the channels is faulty, ...)
%lxy.i.ERR	Channel error bit	This bit set at 1 indicates a faulty channel.
Key		
(1)	Address xy.i <ul style="list-style-type: none"><li>● x: corresponds to the rack number</li><li>● y: corresponds to the module number</li><li>● i: corresponds to the channel number</li></ul>	

---

### Word objects

The table below shows the different word objects for implicit exchange.

Object (1)	Function	Meaning
%lWxy.i.0	Signals on input	Common signals (byte 0) <ul style="list-style-type: none"><li>● x0 = 1: DCD signal</li><li>● x1 = 1: RI signal</li><li>● x2 = 1: CTS signal</li><li>● x3 = 1: DSR signal</li></ul>
%QWxy.i.0	Signal on output	<ul style="list-style-type: none"><li>● x0 = rising edge at 1: stops all exchanges in progress</li></ul>
Key		
(1)	Address xy.i <ul style="list-style-type: none"><li>● x: corresponds to the rack number</li><li>● y: corresponds to the module number</li><li>● i: corresponds to the channel number</li></ul>	

---



## Explicit exchange language objects

### Introduction

This page describes all the language objects for explicit exchange in character mode communication that can be displayed or modified by the application program. See Specified exchanges : General points, p. 157.

### Word objects for PCMCIA cards

The table below shows the different word objects for explicit exchange.

Object (1)	Function	Meaning
%MWxy.MOD.2	Module status	<ul style="list-style-type: none"> <li>● x0 = 1: defective module</li> <li>● x1 = 1: functional error (error between the processor and the module, adjustment or configuration error, ...)</li> <li>● x2 = 1: terminal block fault (not connected)</li> <li>● x3 = 1: self-tests running</li> <li>● x4 = 1: reserved</li> <li>● x5 = 1: error in hardware or software configuration (the module present is not that declared in the configuration, the sub-modules are not compatible)</li> <li>● x6 = 1: missing module</li> <li>● x7 = 1: error in one of the sub-modules</li> </ul>
%MWxy.i.2	Standard channel status	<ul style="list-style-type: none"> <li>● x0 = 1: single station on the network</li> <li>● x1 = 1: not used</li> <li>● x2 = 1: transmission line error</li> <li>● x3 = 1: not used</li> <li>● x4 = 1: internal software fault</li> <li>● x5 = 1: error in hard- or software configuration, or configuration missing</li> <li>● x6 = 1: communication error with the processor</li> <li>● x7 = 1: application fault (double address station)</li> </ul>
%MWxy.i.3	Specific channel status	Character mode type = 16#03
%MWxy.i.4	Error counter	Characters sent in error
%MWxy.i.5	Error counter	Characters received in error
%MWxy.i.15	Command	<ul style="list-style-type: none"> <li>● x0 = 1: RAZ counter</li> <li>● x8 = 1: DTR ON signal</li> <li>● x9 = 1: DTR OFF signal</li> </ul>
Key		

Object (1)	Function	Meaning
(1)	Address xy.i <ul style="list-style-type: none"><li>● x: corresponds to the rack number</li><li>● y: corresponds to the module number</li><li>● i: corresponds to the channel number</li></ul>	

---

**Word objects for terminal port**

The table below shows the different word objects for explicit exchange.

Object (1)	Function	Meaning
%MW0.MOD.2	Module status	<ul style="list-style-type: none"><li>● x0 = 1: defective module</li><li>● x1 = 1: functional error (error between the processor and the module, adjustment or configuration error, ...)</li><li>● x2 = 1: terminal block fault (not connected)</li><li>● x3 = 1: self-tests running</li><li>● x4 = 1: reserved</li><li>● x5 = 1: error in hardware or software configuration (the module present is not that declared in the configuration, the sub-modules are not compatible)</li><li>● x6 = 1: missing module</li><li>● x7 = 1: error in one of the sub-modules</li></ul>
%MW0.i.2	Standard channel status	<ul style="list-style-type: none"><li>● x0 = 1: single station on the network</li><li>● x1 = 1: not used</li><li>● x2 = 1: transmission line error</li><li>● x3 = 1: not used</li><li>● x4 = 1: internal software fault</li><li>● x5 = 1: error in hardware or software configuration, or no configuration</li><li>● x6 = 1: communication error with the processor</li><li>● x7 = 1: application fault (double address station)</li></ul>
%MW0.0.3	Specific channel status	Character mode type = 16#03

---

## Explicit exchange management and reports

### Introduction

This page describes all the language objects that manage explicit exchanges. Exchange and report management, p. 159

### Word objects

The table below shows the different word objects for the management of explicit exchanges.

Object (1)	Function	Meaning
%MWxy.MOD.0	Module exchanges in progress	<ul style="list-style-type: none"> <li>● x0 = 1: reading status in progress</li> <li>● x1 = 1: sending of command parameters to the communication module</li> <li>● x2 = 1: sending of adjustment parameters to the communication module</li> </ul>
%MWxy.MOD.1	Module report	<ul style="list-style-type: none"> <li>● x1 = 0: command parameters received and accepted by the module</li> <li>● x2 = 0: adjustment parameters received and accepted by the module</li> </ul>
%MWxy.i.0	Channel exchanges in progress	<ul style="list-style-type: none"> <li>● x0 = 1: reading status in progress</li> <li>● x1 = 1: sending of command parameters to the communication channel</li> <li>● x2 = 1: sending of adjustment parameters to the communication channel</li> </ul>
%MWxy.i.1	Channel report	<ul style="list-style-type: none"> <li>● x1 = 0: command parameters received and accepted by the communication channel</li> <li>● x2 = 0: adjustment parameters received and accepted by the communication channel</li> </ul>
Key		
(1)	Address xy.i <ul style="list-style-type: none"> <li>● x: corresponds to the rack number</li> <li>● y: corresponds to the module number</li> <li>● i: corresponds to the channel number</li> </ul>	

## Language objects associated with configuration

---

### Introduction

This page describes all the configuration language objects for character mode communication that can be displayed by the application program.

---

### Internal constants

The following table describes the internal constants:

Object	Function	Meaning
%KWxy.i.0	Type	Character mode function = 16#03
%KWxy.i.1	Speed / Format	Byte 0: speed <ul style="list-style-type: none"><li>● 00 = 1200 bits/s, ..., 04 = 19200 bits/s</li></ul> Byte 1: format <ul style="list-style-type: none"><li>● x8: bit number (1 = 8 bits, 0 = 7 bits)</li><li>● x9 = 1: parity management</li><li>● x10: Parity type (1 = odd, 0 = even)</li><li>● x11: stop bit (1 = 1 bit, 0 = 2 bits)</li></ul>
%KWxy.i.2	Stop on silence	Value in ms (0 = not active)
%KWxy.i.3	Various	<ul style="list-style-type: none"><li>● x0 = 1: echo on reception</li><li>● x1 = 1: restart echo on 1st character</li><li>● x2 = 1: automatic transmission of L</li><li>● x3 = 1: backspace management</li><li>● x4 = 1: Xon-Xoff flow control active</li><li>● x5 = 1: RTS/DCD flow control active</li><li>● x6 = 1: beep management</li><li>● x7 = 1: RTS/CTS flow control active</li></ul>
%KWxy.i.4	Signal management	<ul style="list-style-type: none"><li>● x0...x7: reserved</li><li>● x8 = 1 if PSR signal management (TSX SCP 112)</li><li>● x9 = 1 if Full Duplex management</li><li>● x10 = 1 if DCD data carrier management (TSX SCP 111)</li></ul>
%KWxy.i.5	RTS/CTS delay	Value of delay in hundreds of ms (default value = 0ms)
%KWxy.i.6	Stop on reception character 1	<ul style="list-style-type: none"><li>● x0 = 1: end character 1 enabled</li><li>● x1 = 1: end character 1 included</li></ul> Byte 1: value of end character in decimal
%KWxy.i.7	Stop on reception character 2	<ul style="list-style-type: none"><li>● x0 = 1: end character 2 enabled</li><li>● x1 = 1: end character 2 included</li></ul> Byte 1: value of end character in decimal

---

**Internal constants for the terminal port**

The following table describes internal constants for the terminal port configured in character mode:

Object	Function	Meaning
%KW0.0.0	Type	Character mode function = 16#03
%KWxy.i.1	Speed / Format	Byte 0: speed ● 00 = 1200 bits/s, ..., 04 = 19200 bits/s Byte 1: format ● x8: bit number (1 = 8 bits, 0 = 7 bits) ● x9 = 1: parity management ● x10: Parity type (1 = odd, 0 = even) ● x11: stop bit (1 = 1 bit, 0 = 2 bits) ● x12 = 1: echo on reception ● x13 = 1: beep management ● x14 = 1: backspace management



---

# Communication via Uni-telway bus

## IV

---

### Introduction

#### Subject of this part

This part introduces the principles of configuring and using Uni-telway communication via PL7 software.

#### What's in this part?

This part contains the following chapters:

Chapter	Chaptername	Page
16	General	265
17	Configuring a Uni-telway communication	273
18	Programming a Uni-telway communication	283
19	Debugging a Uni-telway communication	309
20	Language objects associated with Uni-telway communication	317





Introduction

Subject of Chapter

This Chapter introduces communication via the Uni-telway bus and its services.

What's in this chapter?

This chapter contains the following topics:

Topic	Page
Introduction	266
Compatibility	267
Performance	269
Operating mode	271
Addresses of a slave PLC	272

## Introduction

---

### Introduction

Communication via Uni-telway is used to exchange data between all the devices connected to the bus. The Uni-telway standard is a UNI-TE protocol which creates a hierarchical structure ( i.e. a master and several slaves). The master device is the bus manager

Uni-telway allows devices to communicate on an equal footing, and authorizes messages to be sent from:

- master to slave,
- slave to master,
- slave to slave.

### Associated manuals

If you require further information you should consult the following manuals:

Title	Description
Uni-telway Bus Communication – User Guide	Detailed description of communication using Uni-telway.
X-WAY Communication – Reference Manual	Detailed description of UNI-TE messaging
TSX Micro PLCs - Installation manual	Hardware installation
Premium TSX PLCs - Installation manual	Hardware installation

---

## Compatibility

---

### Hardware

This type of communication is available for:

- TSX Micro and TSX Premium PLCs via the Terminal port associated with the RS485 physical layer,
  - TSX Premium via:
    - a PCMCIA TSX SCP 111 card associated with the RS232 physical layer,
    - a PCMCIA TSX SCP 112 card associated to 20 mA current loops,
    - a PCMCIA TSX SCP 114 card associated with physical layers RS422 and RS485,
    - a Built-in Link with TSX SCY 21600 / 21601 modules, associated with physical layer RS485,
  - TSX Micro PLCs supporting the PCMCIA cards mentioned above.
- 

### Software

The Terminal port on TSX Premium and TSX Micro processors allows:

- processing in Uni-Telway mode, mastering:
  - 4 messages transmitted to the bus,
  - 4 received messages,
- processing in Uni-Telway mode, as a slave of:
  - 4 transactions at server address Ad0,
  - 4 transactions at server address Ad1,
  - 4 receptions at application address Ad2.

For communication via a Terminal port, the maximum frame size is 128 bytes per communication function.

PCMCIA cards and the link built into TSX SCY 21600/21601 modules authorizes:

- processing in Uni-Telway mode, mastering:
  - 8 messages transmitted to the bus,
  - 8 received messages,
- processing in Uni-Telway mode, as a slave of:
  - 6 transactions at server address Ad0,
  - 1 transactions at server address Ad1,
  - 8 receptions at application address Ad2.

For communication via a PCMCIA card or built in link, the maximum frame size is 240 bytes per communication function.

The `READ_VAR` communication function can read up to 1000 consecutive bits in any remote device. To read in excess of 1000 bits, the `SEND_REQ` communication function must be used.

**Note:** TSX Nano, TSX Micro and TSX Premium PLCs cannot send more than 1000 bits after a read request.

---

## Performance

### Introduction

The following tables are used to assess the typical exchange time in Uni-telway mode for:

- PCMCIA cards and the built-in link on TSX SCY 21600/21601 modules,
- the terminal port.

The results shown correspond to the average time (in ms) for the `READ_VAR` function to be executed.

### Time with PCMCIA cards

Number of objects read: 1 word

Speed in bits/s	T cycle in ms	Average time TSX SCP 114	Average time TSX SCY 21600/21601
4800	cyclic	131	152
4800	10	160	172
4800	50	180	200
9600	cyclic	95	110
9600	10	107	120
9600	50	167	190
19200	cyclic	64	84
19200	10	67	87
19200	50	107	130

Number of objects read: 100 words

Speed in bits/s	T cycle in ms	Average time TSX SCP 114	Average time TSX SCY 21600/21601
4800	cyclic	620	638
4800	10	640	660
4800	50	710	730
9600	cyclic	363	387
9600	10	373	395
9600	50	402	428
19200	cyclic	213	230
19200	10	214	240
19200	50	249	272

**Time with  
terminal port**

**Exchange times for TSX Micro PLCs**

Transmission speed = 9600 bits/s, number of objects read = 40 words

T cycle in ms	Average time
10	205
20	213
50	258
100	299
255	457

**Exchange times for TSX Premium PLCs**

Transmission speed = 19200 bits/s, number of objects read = 40 words

T cycle in ms	Average time
10	135
20	150
50	185
100	210
255	340

**Recommendations for use**

To improve the performance of a slave device connection phase on Uni-telway, you are recommended to configure the number of slaves in relation to the number of slaves present, and to choose addresses starting from 1.

---

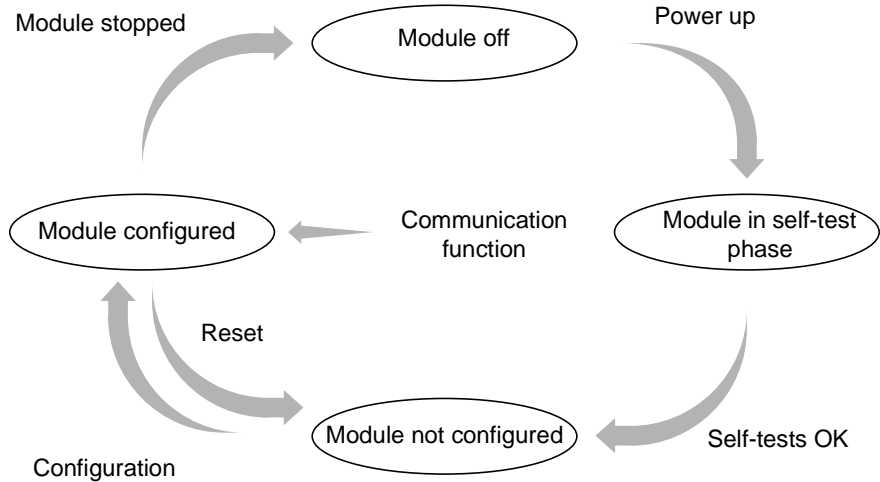
## Operating mode

### Introduction

The following graph describes the operating modes for Uni-Telway PCMCIA cards, for the built-in link on TSX SCY 21600/21601 modules and the terminal port.

### General graph

The operating mode is as follows:



### Operation

- after power-up the module self-tests. During this phase the warning indicators flash.
- if there is no PL7 application in the PLC, the module remains in await configuration mode.
- if there is a PL7 application in the PLC, the application configuration is sent to the module, which then starts.
- in case of power outage, the PLC processor performs a warm restart. The module then re-launches its self-test procedure.

## Addresses of a slave PLC

---

### At a Glance

A slave PLC can have up to three Uni-telway addresses:

- a server address **Ad0**,
- a client application address **Ad1**,
- a listening application address **Ad2**.

---

### Address Ad0

A server address, called **Ad0**, is obligatory and coded in the configuration. It enables 'access to the PLC system' for adjustment, diagnostic or reading functions or 'writing variables, program loading and unloading,...')

---

### Address Ad1

A client application address, called **Ad1**, is supplied optionally by the slave coupler configuration. This enables 'requests or messages requiring a response or not to be sent to another device connected on the Uni-telway bus.

---

### Address Ad2

A listening application address, called **Ad2**, is supplied optionally by the slave coupler configuration. This enables Unsolicited Data (16#FC) requests to be received from 'another device connected on the Uni-telway bus.

---

### Usage constraints

Addresses Ad1 and Ad2 are consecutive to the address Ad0 ( $Ad1 = Ad0 + 1$  and  $Ad2 = Ad0 + 2$ ).

### Example

Uni-telway liaison address	Logical entities	
Ad0 = 6	System	responds to questions
Ad1 = 7	Client application	sends questions to a Uni-telway server device
Ad2 = 8	Listening application	receives the request "Unsolicited Data" made to the application

**Note:** when the Uni-telway master is an SCM (series 7 PLCs), the application contained in the master must use the destination slave address (TSX Micro, Premium) increased by 100 (16#0064).

---



---

# Configuring a Uni-telway communication

17

---

## Introduction

### Subject of this Chapter

This Chapter describes the Configuration process for setting up a Uni-telway communication.

### What's in this chapter?

This chapter contains the following topics:

Topic	Page
How to access Uni-telway PCMCIA card parameters	274
How to access terminal port parameters	275
How to access TSX SCY 21600/21601 module parameters	276
Uni-telway link configuration screen	277
Functions accessible using Uni-telway	278
Uni-telway parameters linked to the application	279
Uni-telway parameters linked to transmission	281

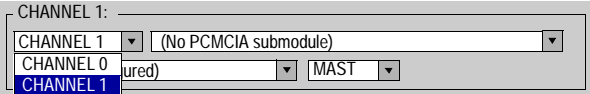
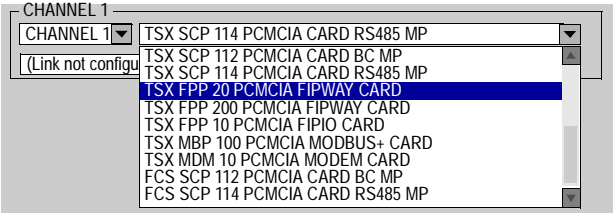
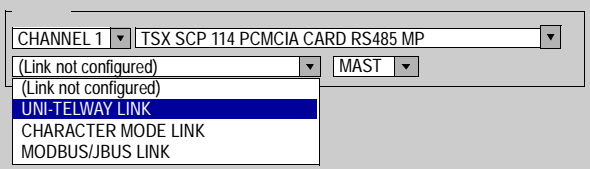
## How to access Uni-telway PCMCIA card parameters

### Introduction

This operation describes how to access configuration parameters of the Uni-telway link via PCMCIA cards for TSX Premium PLCs.

### How to access the link

The following table shows the procedure for accessing the Uni-telway link:

Step	Action
1	Access the communication channel configuration screen.
2	Select the communication channel from the drop-down menu <b>CHANNEL 1</b> <b>Example</b> 
3	Select one of the following PCMCIA cards from the drop-down menu: <ul style="list-style-type: none"><li>● <b>TSX SCP 111 PCMCIA CARD RS232 MP</b></li><li>● <b>TSX SCP 112 PCMCIA CARD BC MP</b></li><li>● <b>TSX SCP 114 PCMCIA CARD RS485 MP</b></li></ul> <b>Example</b> 
4	Select the link from the drop-down menu <b>UNI-TELWAY LINK:</b> <b>Example</b> 

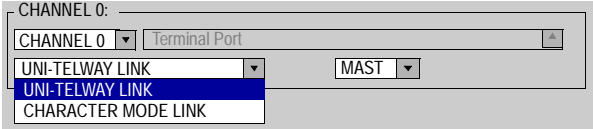
## How to access terminal port parameters

### Introduction

This operation describes how to access the configuration parameters of the Uni-telway link via the TSX Micro PLC terminal port.

### How to access the link

The following table shows the procedure for accessing the Uni-telway link:

Step	Action
1	Access the communication channel configuration screen.
2	Select the link from the drop-down menu <b>UNI-TELWAY LINK:</b> <b>Example</b> 

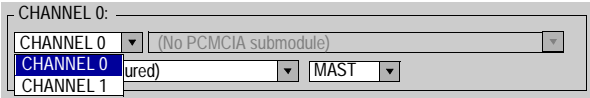
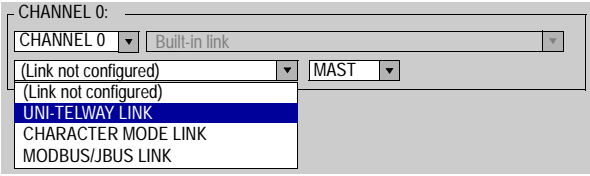
## How to access TSX SCY 21600/21601 module parameters

### Introduction

This operation describes how to access the configuration parameters of the Uni-telway link via TSX SCY 21600 / 21601 modules for TSX Premium.

### How to access the link

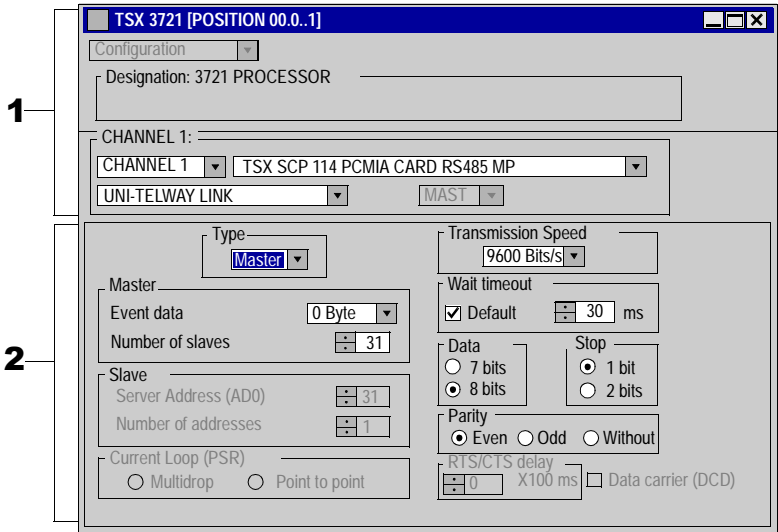
The following table shows the procedure for accessing the Uni-telway link:

Step	Action
1	Access the communication channel configuration screen of the chosen module
2	Select the communication channel from the drop-down menu <b>CHANNEL 0</b> <b>Example</b> 
3	Select the link from the drop-down menu <b>UNI-TELWAY LINK:</b> <b>Example</b> 

## Uni-telway link configuration screen

**Introduction** This screen, split into two zones, is used to declare the communication channel and to configure the necessary parameters for a Uni-telway link.

**Illustration** The screen dedicated to Uni-telway communication looks like this:



**Elements and functions** This table describes the different zones that make up the configuration screen:

Address	Zone	Function
1	common	See Description of configuration screens for communication, p. 168.
2	specific	Is used to select or complete the parameters of a Modbus link. It is split into two types of information: <ul style="list-style-type: none"><li>the application parameters,</li><li>the transmission parameters.</li></ul>

## Functions accessible using Uni-telway

---

### Introduction

Depending on the chosen communication supports, some parameters cannot be modified. They appear grayed out.

---

### Accessible functions

The summary table below shows the various choices possible:

Functions	SCP 111	SCP 112	SCP 114	SCY 21600/21601	Terminal Port
Master - Event data	Yes	Yes	Yes	No	No
Master - Number of slaves	Yes	Yes	Yes	Yes	Yes
Slave	Yes	Yes	Yes	Yes	Yes
Current Loop (PSR)	No	Yes	No	No	No
Transmission speed	Yes	Yes	Yes	Yes	Yes
Wait timeout	Yes	Yes	Yes	Yes	Yes
Data / Stop	Stop	Stop	Stop	Stop	No
Parity	Yes	Yes	Yes	Yes	Yes
RTS/CTS delay	Yes	No	No	No	No
Carrier management (DCD)	Yes	No	No	No	No

---

---

## Uni-telway parameters linked to the application

---

### Introduction

Once the communication channel has been configured, complete the parameters dedicated to the application.

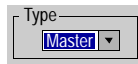
They are split into four windows:

- the **Type** window,
- the **Master** window,
- the **Slave** window,
- the **Current loop (PSR)** window.

---

### Parameter type

The window looks like this:



It is used to select the type of Uni-telway protocol used by the module:

- **Master**: selects the choice of Uni-telway master,
- **Slave**: selects the choice of Uni-telway slave.

---

### Master function

The window can only be accessed by selecting **Master** as the type:

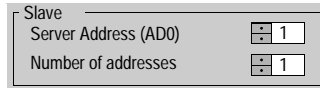


It can be used to enter:

- **Event data**: is used to choose the number of bytes for Event data:
  - the default value is 0 bytes,
  - values are 0, 4 or 8 bytes.
- the **Slave Number**: is used to select the number of slaves to be scanned by the master PLC:
  - values are between 0 and 98,
  - the default value depends on the communication channel: 31 for a PCMCIA card and built-in link, and 8 for the terminal port.

**Slave function**

The window can only be accessed by selecting **Slave** as the type:

A screenshot of a configuration window titled "Slave". It contains two fields: "Server Address (Ad0)" with a value of 1, and "Number of addresses" with a value of 1. Both fields have up and down arrow buttons.

It can be used to enter:

- the **Server Address (Ad0)**: is used to choose the device's server address Ad0,
  - values are between 1 and 98 (for a PCMCIA card and built-in link), or between 1 and 8 (for the terminal port),
  - the default value is 1.
- the **Address Number**: is used to assign up to three slave addresses to one device. This option is available, for example, to programmable controllers which can use Server addresses (Ad0), Client addresses (Ad1), and Listen application addresses (Ad2),
  - values are between 1 and 3 (1 for Ad0 only, 2 for Ad0 and Ad1, 3 for Ad0, Ad1 and Ad2),
  - the default value is 1.

---

**Current loop function**

The window looks like this:

A screenshot of a configuration window titled "Current Loop (PSR)". It contains two radio buttons: "Multidrop" and "Point to point". The "Point to point" radio button is selected.

It is used to select a type of communication:

- **Multidrop** (in current loop),
  - **Point to point** (in current loop).
-



---

## Uni-telway parameters linked to transmission

---

### Introduction

Once the communication channel has been configured, complete the parameters dedicated to transmission.

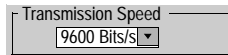
They are split into six windows:

- the **Transmission speed** window,
- the **Wait timeout** window,
- the windows specific to **Data** and **Stop**,
- the **Parity** window,
- the **RTS/CTS delay** window.

---

### Transmission speed

The window looks like this:



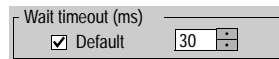
It is used to select the transmission speed for the Uni-telway protocol used by the module:

- the default speed is 9600 bits/s,
- the available speeds are 1200, 2400, 9600 and 19200 bits/s,
- speeds of 300 and 600 bits/s are only available with the TSX SCP 111 PCMCIA card.

---

### Wait timeout

The window looks like this:



This parameter can be used to set the timeout in milliseconds, at the end of which the destination station will be considered missing if there is no response.

- values are between 1 and 255 (for the terminal port), or between 1 and 65000 ms (for a PCMCIA card and built-in link),
- the default value is 30 ms.

---

### Data

The window looks like this:

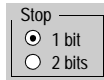


The **Data** field sets the type of coding used to communicate on Uni-telway. All characters are coded on 8 bits.

---

**Stop**

The window looks like this:

A small window titled "Stop" with two radio button options: "1 bit" (selected) and "2 bits".

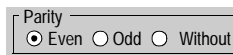
The **Stop** field is used to set the number of stop bits employed for communicating on Uni-telway. Possible values are 1 or 2 stop bits

**Note:** The default value is 1 stop bit.

---

**Parity**

The window looks like this:

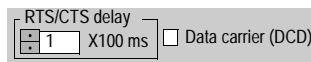
A small window titled "Parity" with three radio button options: "Even" (selected), "Odd", and "Without".

This field is used to define the addition or non-addition of a parity bit, and its type. The possible values are Even, Odd or none (Even by default).

---

**RTS/CTS delay**

The window looks like this:

A window titled "RTS/CTS delay" containing a numeric input field with "1" inside, followed by "X100 ms", and a checkbox labeled "Data carrier (DCD)".

Before each character string transmission, the module activates the RTS signal, and waits for the CTS signal to be activated.

It can be used to enter:

- the maximum waiting time between the two signals. If this time has elapsed, the request is not sent onto the bus.
  - the value is expressed in milliseconds,
  - the default value is 0 ms,
  - the value is between 0s and 10 s,
  - the value 0 specifies the absence of delay management between the two signals.
- data carrier management (DCD signal) in the event of communication with a controlled carrier modem. .
  - if the option is selected, character reception is only enabled if the DCD carrier signal is detected,
  - if the option is not selected, all the characters received are taken into account.

---

# Programming a Uni-telway communication

18

---

## Introduction

### Subject of this Chapter

This Chapter describes the Programming process for setting up a Uni-telway communication.

### What's in this chapter?

This chapter contains the following Sections:

Section	Topic	Page
18.1	Communication function	284
18.2	Master to slave exchange	287
18.3	Slave to master exchange	289
18.4	Slave to slave exchange	295
18.5	Event data	307

# 18.1                      Communication function

---

## Introduction

---

**Subject of section**                      This section introduces the communication functions in Uni-telway mode.

---

**What's in this section?**                      This section contains the following topics:

Topic	Page
Available communication functions	285
Writing command words	286

---

---

## Available communication functions

---

### Introduction

This page describes the communication functions available in Uni-telway mode.

---

### Available functions

Five specific communication functions have been defined to send and receive data to/from a master or slave Uni-telway device:

- **READ\_VAR**: reads a basic language object (e.g. words, bits, double words, floating points, constant words, system words and bits, timer, monostable, drum etc.) See Reading standard objects: **READ\_VAR**, p. 74.
- **WRITE\_VAR**: writes a basic language object (e.g. words, bits, double words, floating points, constant words, system words and bits etc.) See Writing standard objects : **WRITE\_VAR**, p. 87.
- **SEND\_REQ**: exchanges a UNI-TE request. See Transmitting UNI-TE Requests: **SEND\_REQ**, p. 95.
- **DATA\_EXCH**: sends and/or receives text type data. See Exchange of text data : **DATA\_EXCH**, p. 104.
- **MMI functions**: exchanges the different communication functions which are specific to the MMI (**Send\_Msg**, **Send\_alarm**, **Ask\_Msg**, **Ini\_Buttons**, **Control\_Leds**, **Command**).

**Note:** The availability of these functions varies with the type of exchange and the hardware version (refer to the different exchange types).

---

## Writing command words

---

### Introduction

The `WRITE_CMD` instruction is used to explicitly write associated command words in the module, the communication channel, or in the integrated interface.

If there is a Uni-telway link, this instruction will mainly be used while communicating with an external modem.

**Example:** switching of Uni-telway to character mode for the dialing phase.

---

### Syntax

The syntax of the instruction is as follows:

```
WRITE_CMD %CHx.i  
where x: module address and i: channel.
```

---

### Recommendations for use

Before performing a `WRITE_CMD`, you must test whether an exchange is in progress by using a `%MWxy.i.0` language object. To do this, a `READ_STS` operation must be performed to read the word.

Then, the value of the command language object must be modified to carry out the command required. For a Uni-telway link, the language object is the internal word `%MWxy.i.15`.

**Example:** For Uni-telway to switch to character mode, `%MWxy.i.15` is equal to `16#4000` (`%MWxy.i.15:x6 = 1`).

**Note:** A command bit must be switched once from 0 to 1 before sending a `WRITE_CMD`.

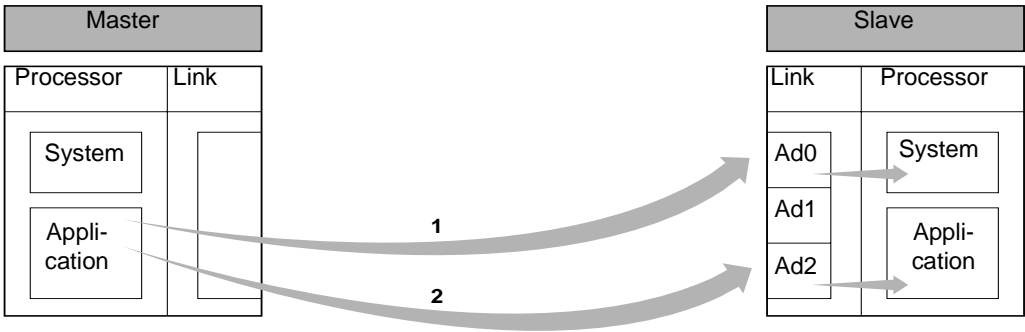
Finally, a `WRITE_CMD` operation must be executed to acknowledge the command.

---

# 18.2 Master to slave exchange

## Master to Slave exchanges

**Introduction**      The master station makes exchanges with the slave station:



### Exchange with address Ad0

The master exchange with Ad0, identified by address mark 1, allows the master application program to communicate with the slave system (access to different objects, etc.).

READ\_VAR, WRITE\_VAR and SEND\_REQ functions can be used to communicate with Ad0.

The function address is of the type `ADR#xy.i.x` where:

Parameters	Description
xy.i	Rack and module number. Channel number
x	Slave address Ad0

### Example

`ADR#0.1.Ad0` for a slave connected to a PCMCIA card on the master PLC.

**Exchange with address Ad2**

Master exchange with Ad2, identified by the address mark 2, is used to send master application program messages to the slave application program.

SEND\_REQ and DATA\_EXCH functions can be used to communicate with Ad2.

The function address is of the type `ADR#xy.i.x` where:

Parameter	Description
xy.i	Rack and module number. Channel number
x	Slave address Ad2

**Example**

```
SEND_REQ(ADR#0.1.Ad2, 16#FC, %MW.....)
```

In this case: use request code, 16#FC, unprompted data.

---



---

# 18.3            Slave to master exchange

---

## Introduction

**Subject of section**            This section introduces slave station to master station exchanges.

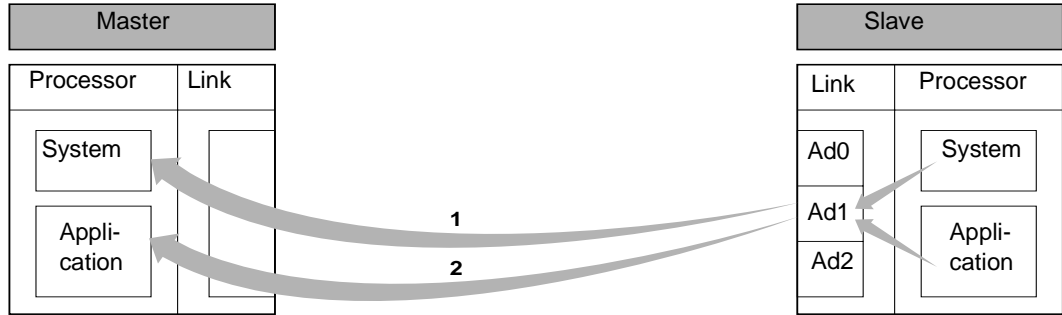
**What's in this section?**            This section contains the following topics:

Topic	Page
Slave to Master exchanges	290
Example of a slave exchange with the master system	292
Example of a direct slave exchange with the master system	294

## Slave to Master exchanges

### Introduction

The slave station makes exchanges with the master station:



### Exchange with the master system

The exchange of slave Ad1 with the master, identified by address mark 1, allows the slave application program to communicate with the master system (access to different objects, etc.).

### Exchange with the application

The exchange of slave Ad1 with the master, identified by the address mark 2, is used to send slave application program messages to the master application program.

### Communication function

Use of the `SEND_REQ` function by the slave requires a 6 byte table to be placed at the start of the transmission buffer which corresponds to the destination address. The first six bytes of the transmission buffer are coded as follows:

	Byte 1 (most significant)	Byte 0 (least significant)
Word 1	station	network
Word 2	module or selector number	gate number
Word 3	reference if gate 8	channel number

To transmit to master system identified by gate 0:

	Byte 1 (most significant)	Byte 0 (least significant)
Word 1	16#FE	16#00
Word 2	16#00	16#00
Word 3	16#00	16#00

To transmit to master application identified by gate 16:

	Byte 1 (most significant)	Byte 0 (least significant)
Word 1	16#FE	16#00
Word 2	16#00	16#10
Word 3	16#00	16#00

**Note:** In the case of a master TSX 47-10, the gate number is 16 + no. of text block

To transmit to a remote PLC system (network 2, station 3):

	Byte 1 (most significant)	Byte 0 (least significant)
Word 1	16#03	16#02
Word 2	16#00	16#00
Word 3	16#00	16#00

## Addressing

When a slave uses the SEND\_REQ function, the syntax is as follows:

SEND\_REQ(ADR#xy.i.x, request number, %MW1:size, ...)

The function address is of the type ADR#xy.i.x where:

Parameter	Description
xy.i	Rack and module number. Channel number
x	Client address AD1 of sender

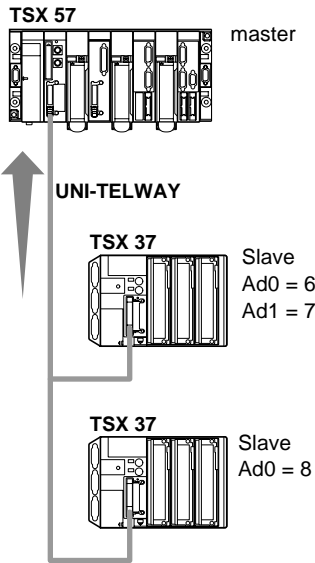
%MW1:size is a word table containing the destination address structured as follows:

If access to the master system	If access to the master application
%MW1 = FE 00	%MW1 = FE 00
%MW2 = 00 00	%MW2 = 00 10
%MW3 = 00 00	%MW3 = 00 00
%MW4 = request parameters:	%MW4 = request parameters:
%MW ... = ...	%MW ... = ...

## Example of a slave exchange with the master system

### Introduction

The slave sends a communication to the master system:



### Transmission

Identification request is sent:

`SEND_REQ(ADR#0.1.7, 15, %MW0:3, %MW10:30, %MW40:4)`

Parameters of request:

Parameters	Description
ADR#0.1.7	<ul style="list-style-type: none"><li>● 0: module</li><li>● 1: channel 1</li><li>● 7: Sender address Ad1</li></ul>
15 or 16 #0F	identification request
%MW0 = 16#FE 00	access to the master system gateway
%MW1 = 16#00 00	
%MW2 = 16#00 00	
%MW43 = 6	3 words sent (= 6 bytes)

Reception

After the exchange:

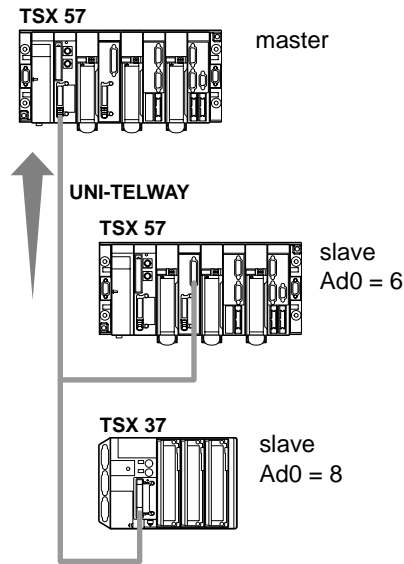
Parameters	Description
%MW40 = 16# 11 00	
%MW41 = 16# 3F 00	16#3F = report >0 (request code + 16#30)
%MW42 = 16# 00 00	
%MW43 = 16# 00 14	14 bytes received from %MW10

## Example of a direct slave exchange with the master system

---

### Introduction

The reception channels of TSX 37 V2.0 processors, and of modules TSX SCY 21600 or TSX SCY 21601 with PCMCIA cards (TSX SCP111, 112, 114 version 1.5) enable `READ_VAR` and `WRITE_VAR` communication functions to be used to communicate with the master server:



### Transmission

Access to the master server from module SCY 21600 / 21601 in slave rack position 0 and via the built-in link:

```
READ_VAR(ADR#2.0.0, '%MW', 0, 5, %MW20:5, %MW50:4)
```

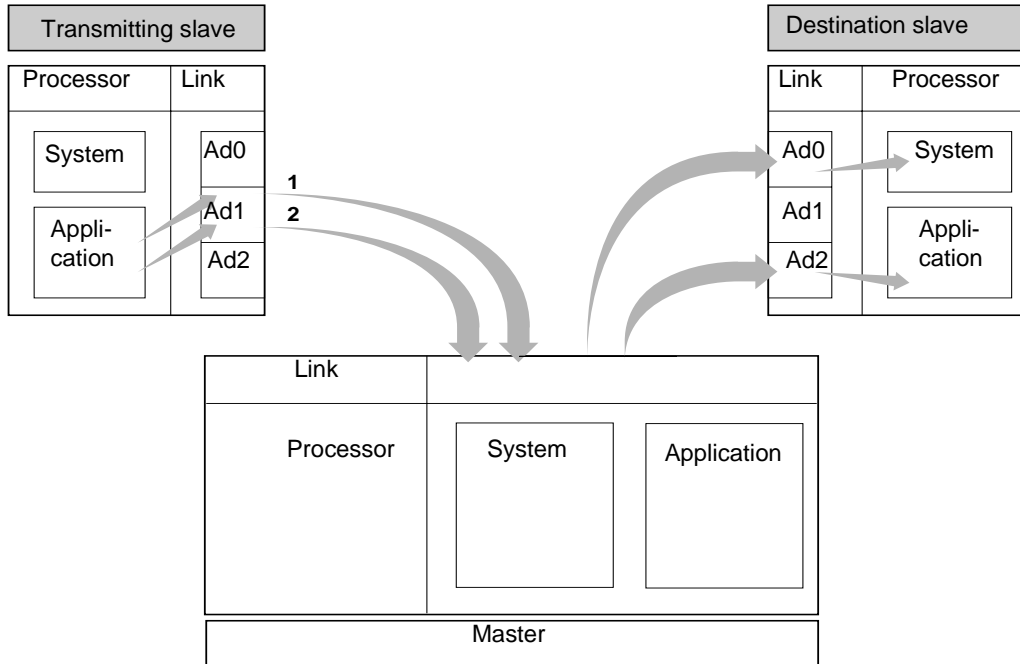
---



## Slave to Slave exchanges

### Introduction

The slave station makes exchanges with another slave station:



### Ad1 to Ad0 exchange

Exchange of slave Ad1 with slave Ad0, identified by address mark 1, allows the sender slave application program to communicate with the destination slave system (access to different objects, etc.).

**Note:** In all cases, the requests transit via the master in total transparency.

### Exchange with the application

Exchange of slave Ad1 with slave Ad2, identified by the address mark 2, is used to send messages from the sender slave application program to the destination slave application program.



**Communication  
function**

Use of the `SEND_REQ` function by the slave requires a 6 byte table to be placed at the start of the transmission buffer which corresponds to the destination address. The first six bytes of the transmission buffer are coded as follows:

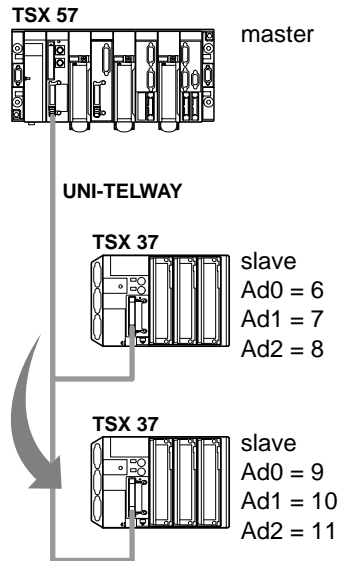
	Byte 1 (most significant)	Byte 0 (least significant)
Word 1	16#FE	16#00
Word 2	16#FE	16#05
Word 3	16#00	number of destination slave (Ad0 or Ad2)

---

## Example of a slave exchange with the slave server

### Introduction

The slave sends a communication function to the slave server:



### Transmission

Writing a 5 word table into slave 9 by slave 6/7/8 from word %MW50:

`SEND_REQ(ADR#0.1.7, 16#0037, %MW100:11, %MW120:1, %MW130:4)`

Parameters of request:

Parameters	Description
ADR#0.1.7	<ul style="list-style-type: none"><li>0: module</li><li>1: channel 1</li><li>7: Sender address Ad1</li></ul>
16 #0037	write objects request
%MW100 = 16#FE 00	address of destination slave (Ad0 = 9)
%MW101 = 16#FE 05	
%MW102 = 16#00 09	
%MW103 = 16#07 68	<ul style="list-style-type: none"><li>type of object = 07 (16 bit integer)</li><li>segment = 68 (internal words)</li></ul>
%MW104 = 50	in decimal, origin of table of words to be written
%MW105 = 05	number of words to be written, in decimal

Parameters	Description
%MW106 to %MW110	content of words to be written in the destination devices
%MW120:1	no response: length: 1 byte
%MW133 = 22	length of data to be sent = 11 words (%MW100 to %MW110) and thus 22 octets

---

## Example of a slave exchange with the slave application

---

**Introduction** The slave sends a communication function to the slave application (Ad2).

---

**Transmission** The sender PLC generates an unprompted data request:  
`SEND_REQ(ADR#0.1.7, 16#00FC, %MW100:10, %MW120:1, %MW130:4)`

Parameters of request:

Parameters	Description
ADR#0.1.7	<ul style="list-style-type: none"><li>● 0: module</li><li>● 1: channel 1</li><li>● 7: Sender address Ad1</li></ul>
16 #0037	unprompted data request
%MW100 = 16#FE 00	address of destination slave (Ad2 = 12)
%MW101 = 16#FE 05	
%MW102 = 16#00 0B	
%MW103 to %MW109	application data to be sent

---

**Reception** The data receiver PLC:  
`DATA_EXCH(ADR#0.1.11, 3, %MW10:1, %MW20:10, %MW100:4)`

Parameters of request:

Parameters	Description
ADR#0.1.7	<ul style="list-style-type: none"><li>● 0: module</li><li>● 1: channel 1</li><li>● 11: address Ad2</li></ul>
3	reception request
%MW20 = 16#FE 00	xx: exchange number of sender function
%MW21 = 16#FE xx	
%MW102 = 16#00 00	
%MW23 to %MW29	application data received

---

## Example 2 of a slave exchange with the slave system

**Introduction** The slave at address Ad1 = 7 reads a 5 word table, using the SEND\_REQ function, in the PLC slave at address Ad0 = 9.

**Transmission** The sender PLC generates a request whose code is 16#0036 (read objects):  
 SEND\_REQ(ADR#0.1.7, 16#0036, %MW200:6, %MW210:6, %MW220:4)

Parameters of request:

Parameters	Description
ADR#0.1.7	<ul style="list-style-type: none"> <li>0: module</li> <li>1: channel 1</li> <li>7: Sender address Ad1</li> </ul>
16 #0036	unprompted data request
%MW200 = 16#FE 00	address of destination slave (Ad0 = 9)
%MW201 = 16#FE 05	
%MW202 = 16#00 09	
%MW203 = 16#07 68	<ul style="list-style-type: none"> <li>type of object = 07 (16 bit integer)</li> <li>segment = 68 (internal words)</li> </ul>
%MW204 = 50	in decimal, origin of table of words to be read
%MW223 = 12	6 words sent (= 12 bytes)

**Note:** At the end of function execution the word length in the report is:  
 %MW223 = 11 (reception of 11 bytes = 10 (5 words) + 1 (object type)).

**Reception table** Table of words read:

	Byte 1	Byte 0
%MW210 =	Least significant bit of first word	07: type of objects read
%MW211 =	Least significant bit of second word	Most significant bit of first word
%MW212 =	Least significant bit of third word	Most significant bit of second word
%MW213 =	Least significant bit of fourth word	Most significant bit of third word
%MW214 =	Least significant bit of fifth word	Most significant bit of fourth word
%MW215 =	not significant	Most significant bit of fifth word

The least significant byte of the first word read contains the type of objects read. As a result, the reception table is thus shifted by 1 byte.

An additional word must therefore be provided for in the reception table. Processing the data requires a processing algorithm for this shift, which is to be provided by the user for TSX 37 PLCs (see example below). For TSX 57 PLCs, this algorithm is provided by the `ROR_ARBI` function.

---

## Example of a right shift of 1 byte in a byte table

### Introduction

Table to be shifted:

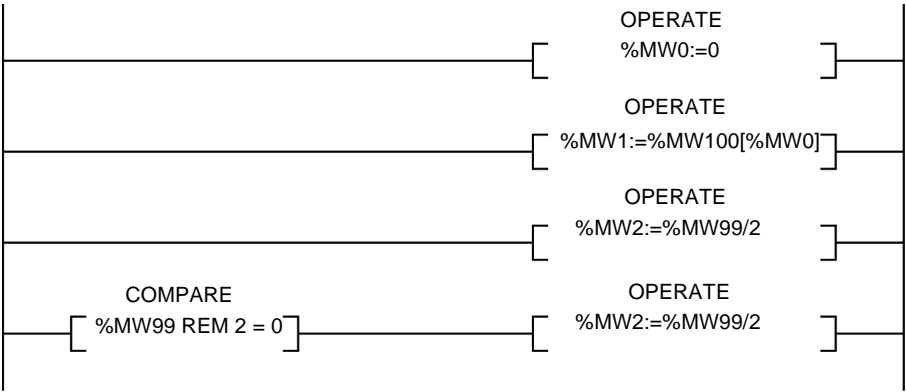
- number of bytes in %MW99,
- starting at %MW100.

Working variables:

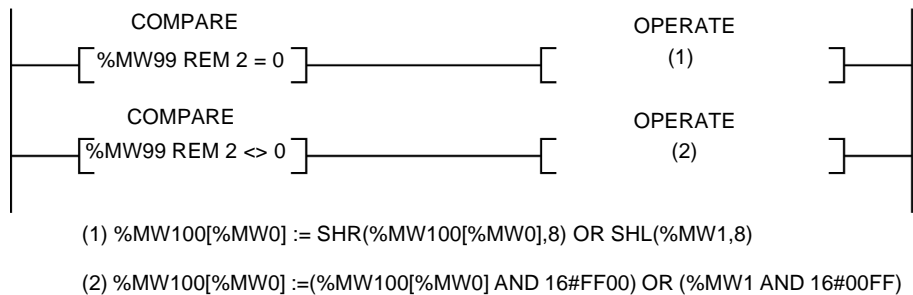
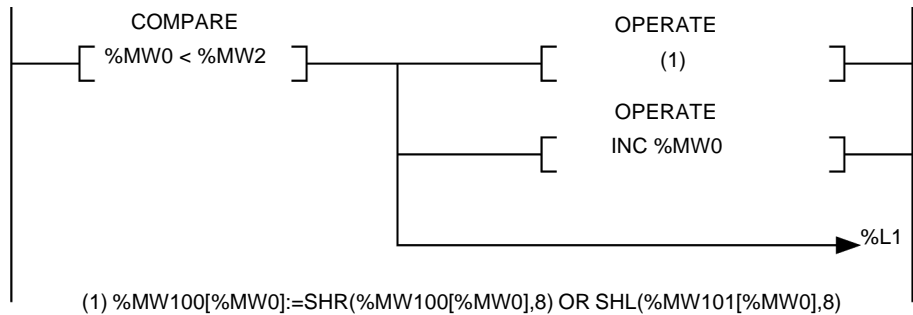
- %MW0,
- %MW1,
- %MW2.

### Program

Initialization of index %MW0 and saving of first word of table in %MW1, %MW2 = length of table -1:



As long as the index is < the length of the table, carry out the shifts:



**Result**

After shifting, the table of words read has the values:

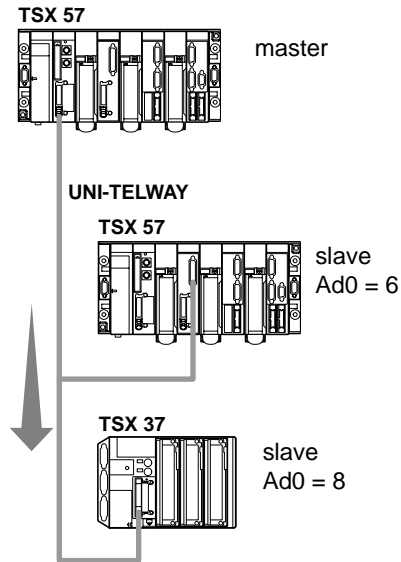
%MW210 =	Most significant bit of first word	Least significant bit of first word
%MW211 =	Most significant bit of second word	Least significant bit of second word
%MW212 =	Most significant bit of third word	Least significant bit of third word
%MW213 =	Most significant bit of fourth word	Least significant bit of fourth word
%MW214 =	Most significant bit of fifth word	Least significant bit of fifth word
%MW215 =	not significant	07: type of objects read



## Example of a direct slave exchange with a slave system

### Introduction

The reception channels for TSX 37 V2.0 processors, and for modules TSX SCY 21600 or TSX SCY 21601 with PCMCIA cards (TSX SCP111, 112, 114 version 1.5) enable `READ_VAR` and `WRITE_VAR` communication functions to be used to communicate with a slave on the same Uni-telway link:



### Transmission

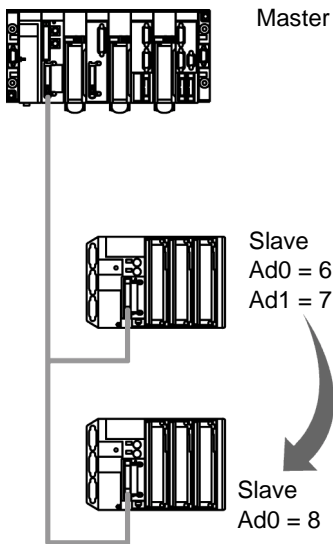
Access to slave server 8 from module SCY 21600 / 21601 in slave rack position 0 and via the built-in link:

```
READ_VAR(ADR#2.0.8, '%MW', 0, 5, %MW20:5, %MW50:4)
```

## Example of a slave stopping another slave

### Introduction

PLC at address Ad0 = 8 STOPPED by PLC at address Ad1 = 7:



### Transmission

`SEND_REQ(ADR#0.1.7, 16#0025, %MW0:3, %MW10:1, %MW40:4)`

Parameters of request:

Parameters	Description
ADR#0.1.7	<ul style="list-style-type: none"><li>0: module</li><li>1: channel 1</li><li>7: Sender address Ad1</li></ul>
16 #0025	STOP request code
%MW0 = 16#FE 00	address of destination slave (Ad0 = 8)
%MW1 = 16#FE 05	
%MW2 = 16#08 00	
%MW43 = 6	length of data to be sent = 3 words, so 6 bytes

## 18.5 Event data

### Event data managed by the master

**Event data** Event data is information sent to the master at the initiative of a server slave station.

**Operating principles** The following table describes the processing phases for communication by event data:

Phases	Description
1	The slave station sends event data to the master station PCMCIA card.
2	The actual reception of data by the card initializes a %IW0.1.2 or %IW0.1.3 word bit. Each input word bit is associated with a link address.
3	When one of these bits has been detected, the application sends a SEND_REQ communication function with the code 16#32 to the master station PCMCIA card to read the data.

**Communication function** Uni-telway request: 16#82 is used to read event data by accessing the Uni-telway PCMCIA server:

```
SEND_REQ(ADR#0.1.SYS, 16#0082, %MW20:10, %MW50:30, %MW100:4)
```

The transmission buffer contains the following data:

	Byte 1 (most significant)	Byte 0 (least significant)
%MW21	16#31	16#06
%MW22	16#01	16#00
%MW23	Slave number	16#00
%MW24	16#FF	16#00
%MW25	16#00	Number of slaves

The buffer corresponds to the following coding:

Parameters	Size	Value
Segment number	1 byte	16#06
Family number	2 bytes	16#0031
Number of type	2 bytes	16#0001
Address	1 byte	Slave number
Access type	2 bytes	16#00FF
Quantity	2 bytes	16#00 No. of slaves



---

# Debugging a Uni-telway communication

19

---

## Introduction

### Subject of this Chapter

This chapter describes the Debugging process during set-up of Uni-telway communication.

### What's in this chapter?

This chapter contains the following topics:

Topic	Page
Uni-telway debugging screen	310
Uni-telway debugging screen	311
Requests available for the communication channel test	312
How to test a channel with Identification and Mirror requests	313
How to test a channel with requests	315

## Uni-telway debugging screen

---

**Introduction** This screen, split into two zones, is used to declare the communication channel and to configure the necessary parameters for communication with nano-PLCs.

---

**Type window** The window looks like this:



**Elements and functions** This table describes the different zones that make up the debugging screen:

Address	Zone	Function
1	common	See Description of communication debugging screens, p. 170.
2	specific	Is used to access link debugging parameters.

---

# Uni-telway debugging screen

**Introduction**

The specific part is split into three windows:

- the **Type** window,
- the **Counters** window,
- the **Channel test** window.

**Type window**

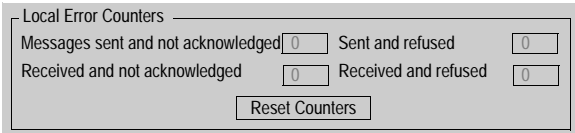
The window looks like this:



It shows the type of Uni-telway function which is configured (master or slave).

**Counters window**

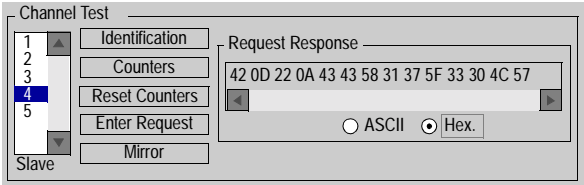
The window looks like this:



This window displays the communication module's different error counters. The button **Reset Counters** resets these counters to zero.

**Channel test window**

The window looks like this:



This window is used to test a communication channel by sending a UNI-TE request to one of the stations on the bus.

## Requests available for the communication channel test

---

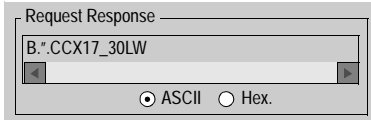
<b>Introduction</b>	This page describes the different possibilities for testing a communication channel from the debugging screen.
<b>Test conditions</b>	<p>Sending a request to an non-server or unconnected slave address results in an error message.</p> <p>When the module has been configured in Uni-telway master mode, the debugging window can be used to send a UNI-TE request to one of the slaves on the bus.</p> <p>When the module has been configured Uni-telway slave mode, the channel test is limited to the master device.</p>
<b>Available requests</b>	<p>The Channel Test window allows the following requests:</p> <ul style="list-style-type: none"><li>● <b>Identification:</b> prompts the Identification request to be sent to the designated slave,</li><li>● <b>Counters:</b> prompts the Read error counters request to be sent to the designated slave,</li><li>● <b>Reset Counters:</b> prompts the designated station's error counters to be reset to zero,</li><li>● <b>Enter request:</b> allows a UNI-TE request, other than those provided by the command buttons, to be sent to the designated slave. Selecting this function gives access to a screen that allows you to select the parameters that are specific to the request (request code must be coded in hexadecimal),</li><li>● <b>Mirror:</b> allows a mirror request to be sent to the designated slave. Selecting this function gives access to a screen that allows you to select the length of the character string to be sent (a maximum of 80 characters ). The PLC then sends this character string (ABCD....) to the destination device. The latter automatically sends the character string that was received back to the sender.</li></ul>



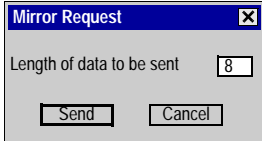
## How to test a channel with Identification and Mirror requests

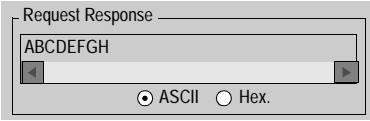
**Introduction** This page indicates the procedure for testing a communication channel by means of Identification and Mirror requests.

**How to identify a station** The following procedure is used to identify a designated station.

Step	Actions
1	Select the server address (Ad0)of the slave to be interrogated using the <b>Slave</b> field.
2	Click on the <b>Identification</b> button.  <b>Result</b> The response appears in the <b>Receive Response</b> window: 

**How to send the Mirror request** The following procedure is used to send the Mirror request and thus to test the routing of information between two devices.

Step	Action
1	Select the server address (Ad0)of the slave to be interrogated using the <b>Slave</b> field.
2	Click on the <b>Mirror</b> button.  <b>Result</b> The following window appears: 
3	Enter the length of data to be sent (maximum 80 characters).

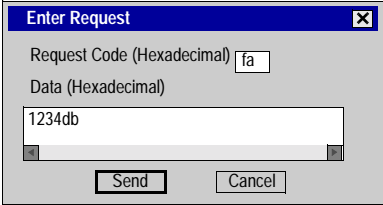
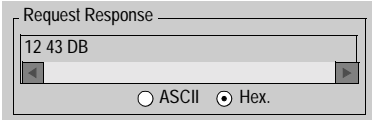
Step	Action
4	<p>Click on the <b>Send</b> button.</p> <p><b>Result</b> The response appears in the <b>Receive Response</b> window:</p> <div data-bbox="676 321 1046 440">A screenshot of a software window titled "Request Response". Inside the window, there is a text box containing the string "ABCDEFGH". Below the text box, there are two radio buttons: "ASCII" (which is selected) and "Hex". The window has a standard Mac OS-style title bar and scroll bars.</div> <p>The response contains:</p> <ul style="list-style-type: none"><li>the character string ABCDEFGH that corresponds to the length of data sent 8.</li></ul>

---

## How to test a channel with requests

**Introduction** This page indicates the procedure for testing a communication channel from the debugging screen using different requests.

**How to send a request** The following procedure is used to send a request, other than those provided by the command buttons, to a designated station.

Step	Action
1	Select the address of the slave to be interrogated using the <b>Slave</b> field.
2	Click on the <b>Enter Request</b> button.  <b>Result</b> The following window appears:   The data sent in this example is coded on 3 bytes.
3	Enter the function code (coded in hexadecimal on one byte), corresponding to the request that you want to send.
4	Enter the data to be sent by coding all the data in hexadecimal. Enter the data continuously without any intervening spaces.
5	Click on the Send button.  <b>Result</b> The response appears in the <b>Receive Response</b> window:   The response from the example has data on 3 bytes (12 43 DB).



---

# Language objects associated with Uni-telway communication

20

---

## Introduction

### Subject of this section

This chapter introduces language objects associated with Uni-telway communication.

### What's in this chapter?

This chapter contains the following Sections:

Section	Topic	Page
20.1	Language objects associated with master Uni-telway mode	318
20.2	Language objects associated with slave Uni-telway mode	325

---

# 20.1                    Language objects associated with master Uni-telway mode

---

## Introduction

---

**Subject of this section**                    This section introduces the language objects associated with operation in master Uni-telway mode.

---

**What's in this section?**                    This section contains the following topics:

Topic	Page
Language object for implicit exchange in master Uni-telway mode	319
Language object for explicit exchange in master Uni-telway mode	320
Exchange management and report	323
Language objects associated with configuration in master Uni-telway mode	324

---

## Language object for implicit exchange in master Uni-telway mode

### Introduction

This page describes all the language objects for implicit exchange in a master Uni-telway communication that can be displayed or modified by the application program. Default exchanges, p. 155

### Bit objects

The table below shows the different bit objects for implicit exchange.

Object (1)	Function	Meaning
%lxy.MOD.ERR	Module error bit	This bit set to 1, indicates a module error (at least one of the channels is faulty,...)
%lxy.i.ERR	Channel error bit	This bit set to 1, indicates a line fault.
Key		
(1)	xy.i Address <ul style="list-style-type: none"> <li>• x: corresponds to the rack number</li> <li>• y: corresponds to the module number</li> <li>• i: corresponds to the channel number</li> </ul>	

### Word objects for PCMCIA cards

The table below shows the different word objects for implicit exchange.

Object (1)	Function	Meaning
%lWxy.i.0	Signals on input	Byte 0: common signals <ul style="list-style-type: none"> <li>• x0 = 1: DCD signal</li> <li>• x1 = 1: RI signal</li> <li>• x2 = 1: CTS signal</li> <li>• x3 = 1: DSR signal</li> </ul>
%lWxy.i.1	General state of slaves	x0 = 1: if at least one slave does not respond
%lWxy.i.2 %lWxy.i.3	State of the event data	1 bit is dedicated to each slave If xi = 1 then the slave at address i has sent data
Key		
(1)	xy.i Address <ul style="list-style-type: none"> <li>• x: corresponds to the rack number</li> <li>• y: corresponds to the module number</li> <li>• i: corresponds to the channel number</li> </ul>	

## Language object for explicit exchange in master Uni-telway mode

### Introduction

This page describes all the language objects for explicit exchange in a master Uni-telway communication that can be displayed or modified by the application program. Specified exchanges : General points, p. 157

### Word objects for PCMCIA cards

The table below shows the different word objects for explicit exchange.

Object (1)	Function	Meaning
%MWxy.MOD.2	Module status	<ul style="list-style-type: none"> <li>● x0 = 1: defective module</li> <li>● x1 = 1: functional error (error between the processor and the module, adjustment or configuration error, ...)</li> <li>● x2 = 1: terminal block fault (not connected)</li> <li>● x3 = 1: self-tests running</li> <li>● x4 = 1: reserved</li> <li>● x5 = 1: error in hardware or software configuration (the module present is not the one declared in the configuration, the sub-modules are not compatible)</li> <li>● x6 = 1: module missing</li> <li>● x7 = 1: error in one of the sub-modules</li> </ul>
%MWxy.i.2	Standard channel status	<ul style="list-style-type: none"> <li>● x0 = 1: no device operating on the channel</li> <li>● x1 = 1: one of the devices on the channel is faulty</li> <li>● x2 = 1: terminal block fault (not connected)</li> <li>● x3 = 1: timeout error (break in cabling, ...)</li> <li>● x4 = 1: self-test running</li> <li>● x5 = 1: hardware or software error or no configuration</li> <li>● x6 = 1: processor communication error</li> <li>● x7 = 1: application fault (configuration error, adjustment error, etc.)</li> </ul>
%MWxy.i.3	Specific channel status	Byte 0 = 0 for the master Uni-telway function
%MWxy.i.4	Error counter	Number of messages sent and not acknowledged
%MWxy.i.5	Error counter	Number of messages sent and refused
%MWxy.i.6	Error counter	Number of messages received and not acknowledged
%MWxy.i.7	Error counter	Number of messages received and refused
%MWxy.i.8 to %MWxy.i.14	State of the slaves	1 bit dedicated to each slave. If xi = 1 then the slave at address i is responding



Object (1)	Function	Meaning
%MWxy.i.15	Command	<ul style="list-style-type: none"><li>● x0 = 1: Reset counter</li><li>● x8 = 1: DTR ON signal</li><li>● x9 = 1: DTR OFF signal</li><li>● x14 = 1: changing from Uni-telway mode to character mode (modem)</li><li>● x15 = 1: changing from character mode (modem) to Uni-telway mode</li></ul>
Key		
(1)	xy.i Address <ul style="list-style-type: none"><li>● x: corresponds to the rack number</li><li>● y: corresponds to the module number</li><li>● i: corresponds to the channel number</li></ul>	

**Word objects for the terminal port**

The table below shows the different word objects for explicit exchange.

Object (1)	Function	Meaning
%MWxy.MO D.2	Module status	<ul style="list-style-type: none"> <li>● x0 = 1: defective module</li> <li>● x1 = 1: functional error (error between the processor and the module, adjustment or configuration error, ...)</li> <li>● x2 = 1: terminal block fault (not connected)</li> <li>● x3 = 1: self-tests running</li> <li>● x4 = 1: reserved</li> <li>● x5 = 1: error in hardware or software configuration (the module present is not the one declared in the configuration, the sub-modules are not compatible)</li> <li>● x6 = 1: module missing</li> <li>● x7 = 1: error in one of the sub-modules</li> </ul>
%MWxy.0.2	Standard channel status	<ul style="list-style-type: none"> <li>● x0 = 1: no device operating on the channel</li> <li>● x1 = 1: one of the devices on the channel is faulty</li> <li>● x2 = 1: terminal block fault (not connected)</li> <li>● x3 = 1: timeout error (break in cabling, ...)</li> <li>● x4 = 1: self-test running</li> <li>● x5 = 1: hardware or software error or no configuration</li> <li>● x6 = 1: processor communication error</li> <li>● x7 = 1: application fault (configuration error, adjustment error, etc.)</li> </ul>
%MWxy.0.3	Specific channel status	Byte 0 = 0 for the master Uni-telway function
%MWxy.0.4	State of the slaves	1 bit dedicated to each slave. If xi = 1 then the slave at address i is responding
Key		
(1)	xy Address <ul style="list-style-type: none"> <li>● x: corresponds to the rack number</li> <li>● y: corresponds to the module number</li> </ul>	

## Exchange management and report

### Introduction

This page describes all the language objects that manage explicit exchanges. Exchange and report management, p. 159

### Word objects

The table below shows the different word objects for the management of explicit exchanges.

Object (1)	Function	Meaning
%MWxy.MOD.0	Module exchanges in progress	<ul style="list-style-type: none"> <li>● x0 = 1: reading status in progress</li> <li>● x1 = 1: sending of command parameters to the communication module</li> <li>● x2 = 1: sending of adjustment parameters to the communication module</li> </ul>
%MWxy.MOD.1	Module report	<ul style="list-style-type: none"> <li>● x1 = 0: command parameters received and accepted by the module</li> <li>● x2 = 0: adjustment parameters received and accepted by the module</li> </ul>
%MWxy.i.0	Channel exchanges in progress	<ul style="list-style-type: none"> <li>● x0 = 1: reading status in progress</li> <li>● x1 = 1: sending of command parameters to the communication channel</li> <li>● x2 = 1: sending of adjustment parameters to the communication channel</li> </ul>
%MWxy.i.1	Channel report	<ul style="list-style-type: none"> <li>● x1 = 0: command parameters received and accepted by the communication channel</li> <li>● x2 = 0: adjustment parameters received and accepted by the communication channel</li> </ul>
Key		
(1)	xy.i Address <ul style="list-style-type: none"> <li>● x: corresponds to the rack number</li> <li>● y: corresponds to the module number</li> <li>● i: corresponds to the channel number</li> </ul>	

## Language objects associated with configuration in master Uni-telway mode

---

### Introduction

This page describes all the configuration language objects for a master Uni-telway communication that can be displayed by the application program.

---

### Internal constants for PCMIA cards

The following table describes the internal constants:

Object	Function	Meaning
%KWxy.i.0	Type	Byte 0 = 0 for the master Uni-telway function
%KWxy.i.1	Speed / Format	Byte 0: speed <ul style="list-style-type: none"><li>● 16#50 = 300 bits/s, 16#51 = 600 bits/s only for TSX SCP 111</li><li>● 16#00 = 1200 bits/s, ..., 16#04 = 19200 bits/s</li></ul> Byte 1: format <ul style="list-style-type: none"><li>● x8: bit number (1 = 8 bits, 0 = 7 bits)</li><li>● x9 = 1: parity management</li><li>● x10: Parity type (1 = odd, 0 = even)</li><li>● x11: stop bit (1 = 1 bit, 0 = 2 bits)</li></ul>
%KWxy.i.2	Wait Timeout	Wait timeout in ms from 5ms to 10,000ms (30ms is the default value)
%KWxy.i.3	Number of slaves	Value from 1 to 98 (31 is the default value)
%KWxy.i.4	Event data size and signal management	Byte 0: event data bytes with a value of 0, 4, or 8 (0 by default) Byte 1 <ul style="list-style-type: none"><li>● x8 = 1 if PSR signal management (TSX SCP 112)</li><li>● x10 = 1 if DCD data carrier management (TSX SCP 111)</li></ul>
%KWxy.i.5	RTS/CTS delay	Value of delay in hundreds of ms (default value = 0ms)

### Internal constants for the terminal port

The following table describes the internal constants:

Object	Function	Meaning
%KWxy.0.0	Type / Speed	Byte 0 = 0 for the master Uni-telway function Byte 1: speed <ul style="list-style-type: none"><li>● 16#00 = 1200 bits/s, ..., 16#04 = 19200 bits/s</li></ul>
%KWxy.0.1	Wait Timeout	Wait timeout in ms from 5ms to 10,000ms (value per 30ms)
%KWxy.0.2	Number of slaves	Value from 1 to 98 (31 is the default value)

---

---

## 20.2 Language objects associated with slave Uni-telway mode

---

### Introduction

#### Subject of this section

This section introduces the language objects associated with operation in slave Uni-telway mode.

#### What's in this section?

This section contains the following topics:

Topic	Page
Language objects for implicit exchange in Uni-telway slave mode	326
Language object for explicit exchange in Uni-telway slave mode	327
Exchange management and report	329
Language objects associated with configuration in slave Uni-telway mode	330

## Language objects for implicit exchange in Uni-telway slave mode

---

### Introduction

This page describes all the language objects for implicit exchange in a slave Uni-telway communication that can be displayed or modified by the application program. Default exchanges, p. 155

---

### Bit objects

The table below shows the different bit objects for implicit exchange.

Object (1)	Function	Meaning
%lxy.MOD.ERR	Module error bit	This bit set to 1, indicates a module error (at least one of the channels is faulty, ...)
%lxy.i.ERR	Channel error bit	This bit set to 1, indicates a line fault.
Key		
(1)	Address xy.i <ul style="list-style-type: none"><li>● x: corresponds to the rack number</li><li>● y: corresponds to the module number</li><li>● i: corresponds to the channel number</li></ul>	

---

### Word objects for PCMCIA cards

The table below shows the different word objects for implicit exchange.

Object (1)	Function	Meaning
%IWxy.i.1	Address state	No master interrogation <ul style="list-style-type: none"><li>● x0 = 1: on Ad0</li><li>● x1 = 1: on Ad1</li><li>● x2 = 1: on Ad2</li></ul>
%IWxy.i.2 %IWxy.i.3	Not significant	-
Key		
(1)	Address xy.i <ul style="list-style-type: none"><li>● x: corresponds to the rack number</li><li>● y: corresponds to the module number</li><li>● i: corresponds to the channel number</li></ul>	

---

## Language object for explicit exchange in Uni-telway slave mode

### Introduction

This page describes all the language objects for explicit exchange in a Uni-telway slave communication that can be displayed or modified by the application program. Specified exchanges : General points, p. 157

### Word objects for PCMCIA cards

The table below shows the different word objects for explicit exchange.

Object (1)	Function	Meaning
%MWxy.MOD.2	Module status	<ul style="list-style-type: none"> <li>● x0 = 1: defective module</li> <li>● x1 = 1: functional error (error between the processor and the module, adjustment or configuration error, ...)</li> <li>● x2 = 1: terminal block fault (not connected)</li> <li>● x3 = 1: self-tests running</li> <li>● x4 = 1: reserved</li> <li>● x5 = 1: error in hardware or software configuration (the module present is not that declared in the configuration, the sub-modules are not compatible)</li> <li>● x6 = 1: missing module</li> <li>● x7 = 1: error in one of the sub-modules</li> </ul>
%MWxy.i.2	Standard channel status	<ul style="list-style-type: none"> <li>● x0 = 1: no device functioning on the channel</li> <li>● x1 = 1: a device on the channel is faulty</li> <li>● x2 = 1: terminal block fault (not connected)</li> <li>● x3 = 1: timeout error (break in cabling, ...)</li> <li>● x4 = 1: self-test running</li> <li>● x5 = 1: error in hardware or software configuration, or no configuration</li> <li>● x6 = 1: communication error with the processor</li> <li>● x7 = 1: application fault (configuration error, adjustment error, etc.)</li> </ul>
%MWxy.i.3	Specific channel status	Byte 0 = 1 for the slave Uni-telway function
Key		
(1)	Address xy.i <ul style="list-style-type: none"> <li>● x: corresponds to the rack number</li> <li>● y: corresponds to the module number</li> <li>● i: corresponds to the channel number</li> </ul>	

**Word objects for the terminal port**

The table below shows the different word objects for explicit exchange.

Object (1)	Function	Meaning
%MWxy.MOD.2	Module status	<ul style="list-style-type: none"> <li>● x0 = 1: defective module</li> <li>● x1 = 1: functional error (error between the processor and the module, adjustment or configuration error, ...)</li> <li>● x2 = 1: terminal block fault (not connected)</li> <li>● x3 = 1: self-tests running</li> <li>● x4 = 1: reserved</li> <li>● x5 = 1: error in hardware or software configuration (the module present is not that declared in the configuration, the sub-modules are not compatible)</li> <li>● x6 = 1: missing module</li> <li>● x7 = 1: error in one of the sub-modules</li> </ul>
%MWxy.0.2	Standard channel status	<ul style="list-style-type: none"> <li>● x0 = 1: no device functioning on the channel</li> <li>● x1 = 1: a device on the channel is faulty</li> <li>● x2 = 1: terminal block fault (not connected)</li> <li>● x3 = 1: timeout error (break in cabling, ...)</li> <li>● x4 = 1: self-test running</li> <li>● x5 = 1: error in hardware or software configuration, or no configuration</li> <li>● x6 = 1: communication error with the processor</li> <li>● x7 = 1: application fault (configuration error, adjustment error, etc.)</li> </ul>
%MWxy.0.3	Specific channel status	Byte 1 = 0 for the slave Uni-telway function
Key		
(1)	xy Address <ul style="list-style-type: none"> <li>● x: corresponds to the rack number</li> <li>● y: corresponds to the module number</li> </ul>	



## Exchange management and report

### Introduction

This page describes all the language objects that manage explicit exchanges. Exchange and report management, p. 159

### Word objects

The table below shows the different word objects for the management of explicit exchanges.

Object (1)	Function	Meaning
%MWxy.MOD.0	Module exchanges in progress	<ul style="list-style-type: none"> <li>● x0 = 1: reading status in progress</li> <li>● x1 = 1: sending of command parameters to the communication module</li> <li>● x2 = 1: sending of adjustment parameters to the communication module</li> </ul>
%MWxy.MOD.1	Module report	<ul style="list-style-type: none"> <li>● x1 = 0: command parameters received and accepted by the module</li> <li>● x2 = 0: adjustment parameters received and accepted by the module</li> </ul>
%MWxy.i.0	Channel exchanges in progress	<ul style="list-style-type: none"> <li>● x0 = 1: reading status in progress</li> <li>● x1 = 1: sending of command parameters to the communication channel</li> <li>● x2 = 1: sending of adjustment parameters to the communication channel</li> </ul>
%MWxy.i.1	Channel report	<ul style="list-style-type: none"> <li>● x1 = 0: command parameters received and accepted by the communication channel</li> <li>● x2 = 0: adjustment parameters received and accepted by the communication channel</li> </ul>
Key		
(1)	xy.i address <ul style="list-style-type: none"> <li>● x: corresponds to the rack number</li> <li>● y: corresponds to the module number</li> <li>● i: corresponds to the channel number</li> </ul>	

## Language objects associated with configuration in slave Uni-telway mode

---

### Introduction

This page describes all the configuration language objects for a slave Uni-telway communication that can be displayed by the application program.

---

### Internal constants for PCMIA cards

The following table describes the internal constants:

Object	Function	Meaning
%KWxy.i.0	Type	Byte 0 = 1 for the slave Uni-telway function
%KWxy.i.3	Slave addresses	Byte 0: value of slave address Ad0 Byte 1: number of consecutive addresses from 1 to 3
%KWxy.i.4	Signal management	Byte 0: reserved Byte 1 <ul style="list-style-type: none"><li>• x8 = 1 if PSR signal management (TSX SCP 112</li></ul>

---

### Internal constants for the terminal port

The following table describes the internal constants:

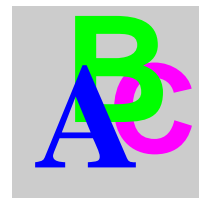
Object	Function	Meaning
%KWxy.0.0	Type / Speed	Byte 0 = 0 for the slave Uni-telway function Byte 1: speed <ul style="list-style-type: none"><li>• 16#00 = 1200 bits/s, ..., 16#04 = 19200 bits/s</li></ul>
%KWxy.0.1	Wait timeout	Byte 0: reserved Byte 1: Wait timeout in ms from 5ms to 10,000ms (value per 30ms)
%KWxy.0.2	Slave addresses	Byte 0: value of slave address Ad0 Byte 1: number of consecutive addresses from 1 to 3

---

---

## Index

---



### Symbols

%CHxy.i, 157

### A

Accessible functions

Character mode, 238

Uni-telway, 278

Accessing configuration

Character mode, 234, 235, 236

Remoting Nano PLCs, 198

Uni-telway, 274

Accessing the configuration editor, 165

Address

Uni-telway, 272

Addressing, 21, 23

### C

CANCEL, 131

Character mode, 221

Characteristics, 146

Character mode, 228

Communication channel test

Uni-telway, 312

Communication functions, 47, 72

Uni-telway, 284

Compatibility

Character mode, 229

Remoting Nano PLCs, 177

compatibility

Uni-telway, 267

Configuration

Character Mode, 233

Remoting of Nano PLCs, 197

Uni-telway, 273

configuration, 163

Configuration parameters

Character mode, 239, 241, 243, 244

Remoting Nano-PLCs, 200

Uni-telway, 279, 281

Configuration screen, 168

Character mode, 237

Remoting Nano PLCs, 199

Uni-telway, 277

Conversion of analog values of input channels, 194

### D

Data exchange, 186

DATA\_EXCH, 104

Debugging

Character mode, 249

Remoting Nano PLCs, 209

Debugging parameters

Character mode, 251

Uni-telway, 310, 311

debugging parameters

Nano PLCs offset, 211

Debugging screen, 170

Character mode, 250

Remoting Nano PLCs, 210

Default exchanges, 155

## E

- Exchange management, 159
  - Character mode, 259
  - Nano remoting, 218
  - Uni-telway, 323, 329
- Exchanging analog data, 189
- Explicit exchange
  - Character mode, 257
  - Nano Deport, 215
  - Uni-telway, 320, 327

## F

- Flow control, 226

## G

- General
  - Character mode, 223
  - Remoting Nano PLCs, 175
  - Uni-telway, 265
- General points, 17

## H

- Help whilst entering, 66
- How to access configuration
  - Uni-telway, 275, 276

## I

- Identification
  - Uni-telway, 313
- Implicit exchange
  - Character mode, 256
  - Nano remoting, 214
  - Uni-telway, 319, 326
- INPUT\_CHAR, 122, 247
- Introduction
  - Character mode, 224, 225
  - Remoting Nano PLCs, 176
  - Uni-telway, 266

## L

- Language objects, 154
  - Character mode, 255
  - Remoting Nano PLCs, 213
  - Uni-telway, 317
- Length, 59

## M

- Management parameters, 55
- Mirror
  - Uni-telway, 313

## O

- Operating mode
  - Character mode, 232
  - Remoting Nano PLCs, 182
  - Uni-telway, 271
- OUT\_IN\_CHAR, 126, 247

## P

- Performance
  - Character mode, 230
  - Remoting Nano PLCs, 179
  - Uni-telway, 269
- Performances, 61
- performances
  - offset Nano PLCs, 178
- Presentation of addressing, 22
- PRINT\_CHAR, 117, 247
- Programming
  - Character mode, 247
  - Remoting Nano PLCs, 203
  - Uni-telway, 283
- Programming terminal, 35

## R

- RCV\_TLG, 114
- READ\_Asyn, 143
- READ\_GDATA, 138
- READ\_VAR, 74
- Remoting Nano PLCs, 173

Report, 56, 159  
    Character mode, 259  
    Nano remoting, 218  
    Uni-telway, 323, 329

Requests  
    Uni-telway, 315

ROR1\_ARB, 134

## S

SEND\_REQ, 95

SEND\_TLG, 111

SERVER, 140

Server, 64

Services

    Remoting Nano PLCs, 185

Specified exchanges, 157

Structure

    Communication functions, 50

SWAP, 137

## T

Testing a communication channel

    Character mode, 253

Timeout, 59

Type

    Addressing, 25

## U

UNI-TE requests, 100

Uni-telway, 263

    Debugging, 309

## W

WRITE\_Asyn, 143

WRITE\_GDATA, 139

WRITE\_VAR, 87

