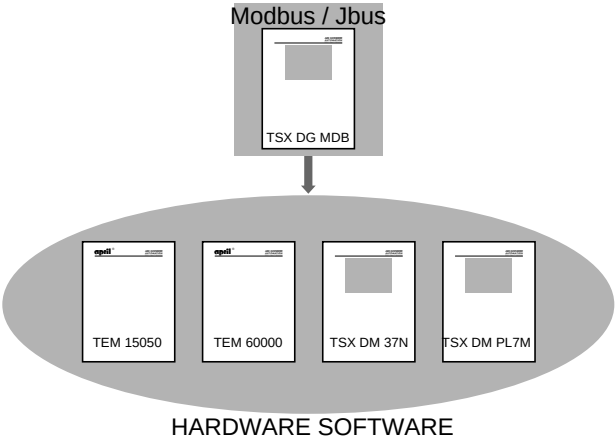
## 1.1 Network documentation presentation

This manual is designed for users implementing a Modbus/Jbus network.

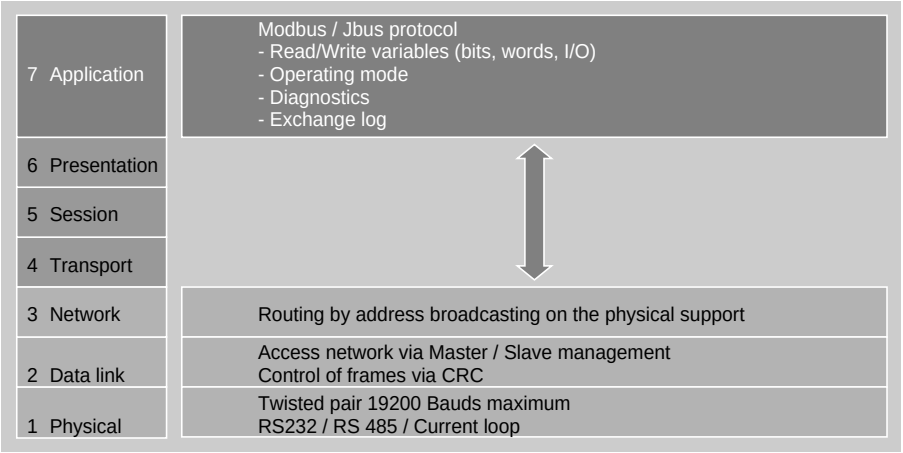
The complete network documentation set is structured as follows :

- general information on X-WAY communication is covered in the TSX DR NET Communication Reference Manual,
- general information on hardware is described in the Users manual : TSX DM 37E,
- general information on the installation of the software for the various networks is given in the manual : TLX DM PL7 M10E,
- information specific to each network is given in the specialized manuals :
  - FIPWAY network : TSX DG FPWE
  - UNI-TELWAY bus : TSX DG UTWE
  - **Modbus/Jbus protocol : TSX DG MDBE (this document)**
  - JBus reference manual : TEM 60000E



## 1.2 Integration in the OSI model

The harmonization of Modbus/Jbus and the OSI model is carried out at the physical, data link and application layers :




Note :
The operating protocols or mechanisms specified in layers 2, 3 and 7 are not standardized and are actual standards.

## 1.3 Presentation

Communication via Modbus is used for exchanging data between all the devices connected on the bus. The Modbus protocol is a protocol which creates a hierarchical structure (one master and several slaves). A muiltidrop link connects the master and the slaves. The master manages all the exchanges exclusively, two types of dialog are possible :

- the master exchanges with one slave and waits for its response,
- the master exchanges with all the slaves without waiting for a response (general broadcast).

This type of communication is available on all PCMCIA format communication modules with integrated Modbus link. This link is associated with the physical layer :

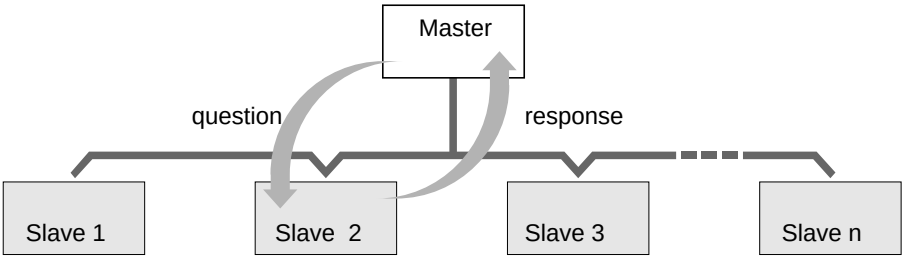
- RS232 for the TSX SCP 111 module,
- 20 mA current loop for the TSX SCP 112 module,
- RS 422 / 485 for the TSX SCP 114 module.

The hardware installation of these cards is detailed in manual TSX DM 37E, part L.

## 1.4 Operating principle :

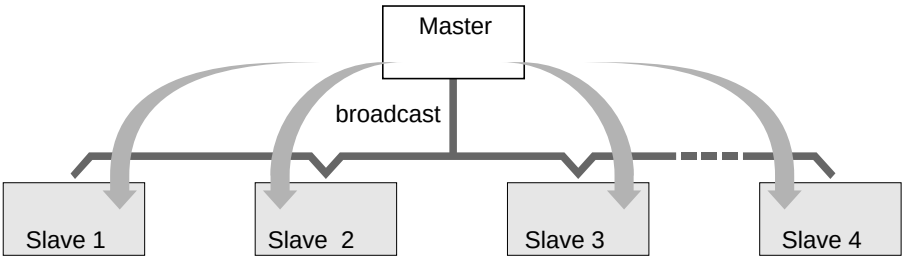
The PLCs use the "master- slave" technique, in which one device (the master) initiates transactions by sending a request. The device to which the request is addressed (slave) sends its response to the request to the master. The master can also broadcast a message to all the slaves (broadcast request), in which case the slaves do not respond to the master.

Question/response mechanism :




The master interrogates a slave with a unique number on the network, and waits for a response from this slave.

Broadcasting mechanism :




The master broadcasts a message to all the slaves present on the network, which execute the order of the message without sending a response.

### 1.4-1 The question :

The question contains a function code indicating to the slave addressed what type of action is requested. The data contains additional information which the slave needs to execute this function. The control bytes field enables the slave to ensure the integrity of the content of the question.

### 1.4-2 The response :

When a slave sends a response following a normal transaction, the function code of the response is an echo of that contained in the question. The data is that collected by the slave, for example the value of a register or a status. If an error appears, the function code is modified to indicate that the response is an error response. The data then contains a code (exception code) enabling the type of error to be recognized. The control field enables the master to confirm that the message is valid.
After receiving a question, a slave station checks the consistency of the frame. If an illegal parameter is detected (function code, address, value) or if the station is not able to execute the request, it returns an exception response in the format below.

Example of exception codes : (from a slave station)

01 : unknown function code,
02 : incorrect address,
03 : incorrect value,
04 : station not ready to execute the request,
05 : acknowledgment, the station has accepted and is processing the request,
06 : the station is processing and unavailable,
07 : negative acknowledgment,

Other exception codes may exist and are specific to each product connected on the network.

### 1.4-3 Format of a question/response frame :

Question :

| Slave number | Function code | Specific information concerning the request (address, number, value, etc) | Control word |
|---|---|---|---|
| 1 byte | 1 byte | n bytes | 2 bytes |

Positive response :

| Slave number | Function code | data received | Control word |
|---|---|---|---|
| 1 byte | 1 byte | n bytes | 2 bytes |

Exception response :

| Slave number | Function code | Exception code | Control word |
|---|---|---|---|
| 1 byte | 1 byte | 1 byte | 2 bytes |

This byte takes the value : function code + most significant bits set to 1

### 1.4-4 General format of a frame

Two types of coding can be used to communicate on a Modbus network. All the devices present on the network must be configured according to the same type.

**ASCII type**
In ASCII mode, all the messages begin with the 'colon' ":" character, and end with 'carriage return - line feed' "CRLF". The characters sent in the other fields are hexadecimal type 0-9, A-F. The devices on the network continuously monitor the arrival of the ":" character, and when it arrives, each device decodes the next field (address field) in order to discover the destination address, and then take into account the next characters if the slave is recognized. The end of the message will be indicated by the "CRLF" characters preceded by the two control characters containing the LRC (Longitudinal Reducing Check).

| START | ADDRESS | FUNCTION | DATA | LRC | END |
|---|---|---|---|---|---|
| 1 Character ":" | 2 Characters | 2 Characters | n Characters | 2 Characters | 2 Characters "CRLF" |

**RTU type**

This is the most frequently used mode and is more efficient than ASCII mode.
In RTU mode, the messages begin with a period of silence on the network of at least 3.5 characters. All the devices present on the network listen to the bus continuously, and decode the first byte in order to discover the destination address, and then take into account the next characters if the slave is recognized. Once the last is character is sent, a silence of at least 3.5 characters indicates the end of the message. A new frame can then be sent.
The characters are hexadecimal type 0-9, A-F. The data contained in the frame must contain all the message, and be sent continuously. The integrity of the message is indicated by the content of the CRC (Cyclical Redundancy Check).

| START | ADDRESS | FUNCTION | DATA | CRC | END |
|---|---|---|---|---|---|
| silence | 1 byte | 1 byte | n bytes | 2 bytes | silence |

Important : the maximum message length is 256 characters.

## 2.1 Services supported by Modbus

Modbus offers 19 different functions. They are characterized by a function code on a byte (in hex). Not all devices support all function codes.

| Code | Nature of Modbus functions | S1000 | TSX 37 | Series 7 |
|---|---|---|---|---|
| **H'01'** | **Read n consecutive output bits** | **X** | **X** | **X** |
| **H'02'** | **Read n consecutive input bits** | **X** | **X** | **X** |
| **H'03'** | **Read n consecutive output words** | **X** | **X** | **X** |
| **H'04'** | **Read n consecutive input words** | **X** | **X** | **X** |
| **H'05'** | **Write 1 output bit** | **X** | **X** | **X** |
| **H'06'** | **Write 1 output word** | **X** | **X** | **X** |
| H'07' | Read exception status | X | X | X |
| H'08' | Access diagnostics counters | X | X | |
| H'09' | Remote up/download and operating modes | | | |
| H'0A' | Operation report request | | | |
| H'0B' | Read event counter | X | X | X |
| H'0C' | Read connection events | X | X | X |
| H'0D' | Remote up/download and operating modes | X | | |
| H'0E' | Operation report request | X | | |
| **H'0F'** | **Write n output bits** | **X** | **X** | **X** |
| **H'10'** | **Write n output words** | **X** | **X** | **X** |
| H'11' | Read identification | | X | X |
| H'12' | Remote up/download and operating modes | | | |
| H'13' | Reset slave after unrecovered error | | | |

The services are classified in three categories :
- write or read words or bits,
- functions for device diagnostics,
- functions for managing the operating modes of a device.

The main functions in bold characters are described in detail in the appendix.

TSX 37 **addressing** :
Word 0 is addressed with address 0, word n is addressed with address n.
Bit 0 is addressed with address 0, bit n is addressed with address n.

See reference manual TEM 60000E for S1000 addressing.

### 2.2-1 Main functions

**Read n output bits**
**Code : 01** This function is used to access output bits which can be read or written, and are defined in the memory of a slave.

**Read n input bits**
**code : 02** This function, which is identical to the previous one and has the same limits, applies to input bits (master can read but not write).

**Read n output words**
**Code : 03** This function is used to access output words which can be read or written, and are defined in the memory of a slave.

**Read n input words**
**code : 04** This function, which is identical to the previous one and has the same limits, applies to input words (master can read but not write).

**Write output bit**
**Code : 05** This function is used to set an output bit which is defined in the memory of a slave to 0 or 1 (only accessible in write).

**Write output word**
**code : 06** This function, which is identical to the previous one and has the same limits, applies to output words (master can read but not write).

**Write n output bits**
**Code : 0F** This function enables the master to write output bits, which can be read or written, to the memory of a slave.

**Write n output words**
**code : 10** This function enables the master to write output words, which can be read or written, to the memory of a slave.

### 2.2-2 Secondary functions

**Read exception status**
**Code : 07** This function provides access to 8 status bits which store certain events in a slave.

**Diagnostics**
**Code : 08** This diagnostics function is used to test the system of communication between a master and a slave, by testing some internal data at the slave level. To do this, a subfunction code is inserted in the frame after the function code on 1 byte.

**Echo**
**Code : 08/00** This diagnostics function requests the interrogated slave to return in full the message sent by the master.

**Restart communication**
**Code : 08/01** This function reinitializes the channel (cancel current messages). The configuration of the channel is maintained.

**Read diagnostics register**
**Code : 08/02** This function is used to access a 16-bit word containing information on the status of the slave.

**Change ASCII delimiter**
**Code : 08/03** In ASCII mode, a byte is replaced by two ASCII characters, representing its hexadecimal coding. The successive messages are separated by a delimiter character, initialized at H'0A' (Line Feed).

**Switch to listen mode**
**Code : 08/04** This function forces a slave to switch to listen only mode (LOM). In this mode, the slave stores the messages which are addressed to it, but does not send any response.

**Reset counters to zero**
**Code : 08/0A** This function resets all the counters of a slave monitoring the exchanges and those of the diagnostics register to zero.

**Read line message counter**
**Code : 08/0B** This function is used to access a 16-bit counter (increments of 0 to FFFF) which totals the number of messages seen on the line and processed by the slave.

**Read checksum error counter**
**Code : 08/0C** This function is used to access a 16-bit counter which totals the number of messages received by the slave with a checksum error.

**Read exception error counter**
**Code : 08/0D** This function is used to access a 16-bit counter which totals the number of exception responses sent by the slave module (after receiving a message whose content is incorrect).

**Read slave message counter**
**Code : 08/0E** This function is used to access a 16-bit counter which totals the number of messages received by the slave, whatever their type.

**Read slave no reply counter**
**Code : 08/0F** This function is used to access a 16-bit counter which totals the number of messages sent by the slave to the master and have received no response.

**Read response counter**
**Code : 08/10** This function is used to access a 16-bit counter which totals the number of correct responses sent by the slaves.

**Read listen mode counter**
**Code : 08/11** This function is used to access a 16-bit counter which totals the number of messages received by a slave.

**Read incorrect character counter**
**Code : 08/12** This function is used to access a 16-bit counter which totals the number of incorrect characters received by a slave.

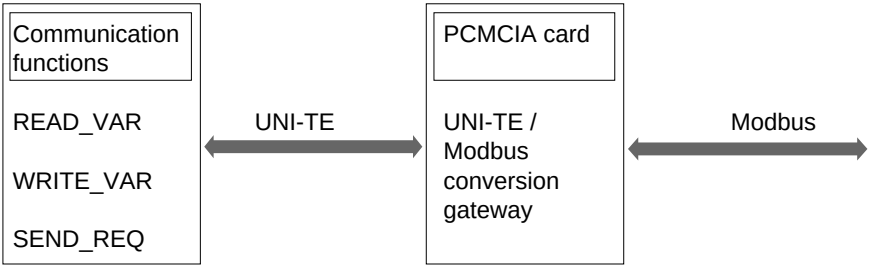
**Read event counter**
**Code : 0B** This function is used to read two 16-bit words : A status and an event counter.

**Read event log**
**Code : 0C** This function is used to access information on a slave :
- status word and event counter (same as function B),
- number of messages seen on the line and processed by the slave (same as function 08/0B),
- contents of connection event counter (64 bytes maximum).

**Read identification**
**Code : 11** This function is used to read a 16-bit word containing status information on the slave addressed.

## 2.3 Functions managed by the PCMCIA card

The PCMCIA card converts the UNI-TE protocol to Modbus protocol. It is thus possible to use the PL7 Micro functions to communicate with Modbus slave devices. The functions used are : READ_VAR, WRITE_VAR, SEND_REQ.

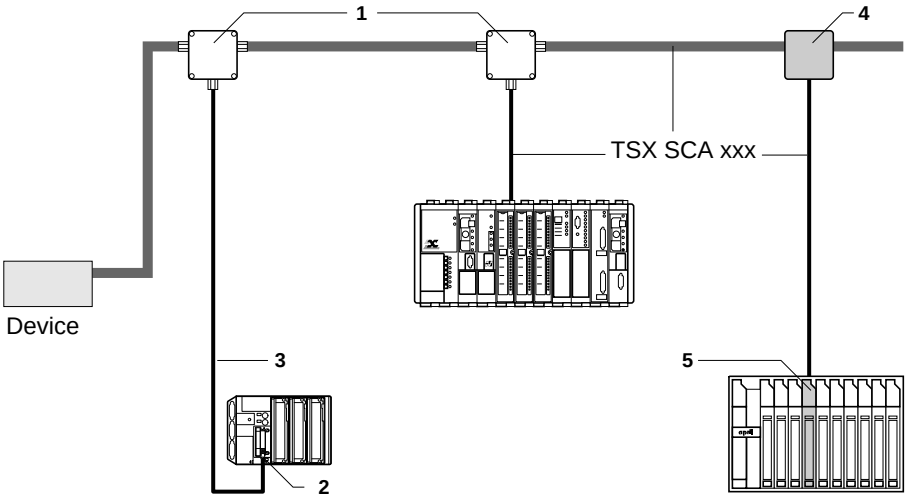



| Modbus function code | PL7 Micro communication function |
|---|---|
| 01 | READ_VAR |
| 02 | SEND_REQ |
| 03 | READ_VAR |
| 04 | SEND_REQ |
| 05 | WRITE_VAR |
| 06 | WRITE_VAR |
| 07 | SEND_REQ |
| 08 + subcodes | SEND_REQ |
| 0B | SEND_REQ |
| 0C | SEND_REQ |
| 0F | WRITE_VAR |
| 10 | WRITE_VAR |
| 11 | SEND_REQ |

The communication functions are described in the PL7 Micro software manual : TLX DS PL7M 10E, volume 2, part L

## 3.1 Hardware installation

Example : RS485 Modbus/Jbus connection




**Presentation of the various elements**

1 **TSX SCA 50 passive T-junction box,** provides impedance matching when installed at the end of a line.

2 **TSX SCP 114 type III PCMCIA card** used for connecting a TSX 37 to the Modbus/ Jbus network via an RS 485 link.

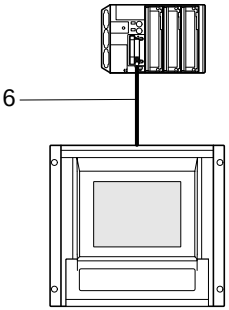
3 **TSX SCP CM 4030 connecting cable** from a TSX SCP 114 PCMCIA card to the Modbus/Jbus network. The length of the cable is 3 m.

4 **TBX 0010 passive T-junction box** for impedance matching at the end of a line and used mainly for connecting series 1000 PLCs to the Jbus network with RS 485.

5 **JBU 0250 or JBU 0550 series 1000 card** used for connecting A5000 and A7000 PLCs to the Jbus network with RS 485.

For more details on the principle of connecting an RS485 link, refer to the following documents :
TSX DRNETE, TSX D41724E, TEM 60000E.

Example : RS 232 Modbus connection



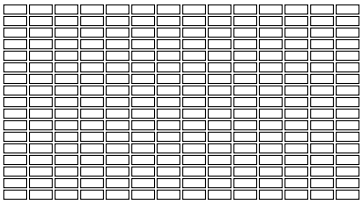

**6** **TSX SCP CD 1030 connecting cable** from a TSX SCP 111 PCMCIA card to a Modbus connected device.

## 3.2 Installing the SCA50

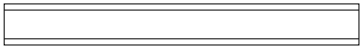
### 3.2-1 Mounting

This can be installed on a pre-slotted mounting plate, ref. AM1 PA... or on a DIN DE/DP Omega rail with LA9 D09976 mounting plate

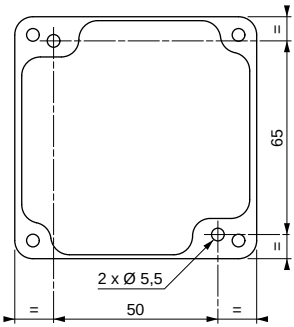
AM1



AM1-PA••



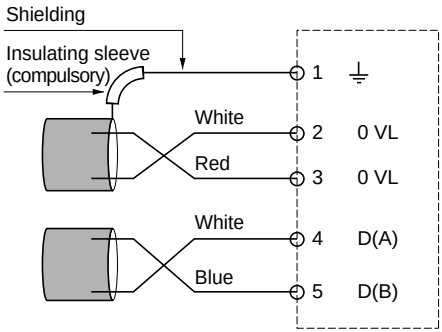
AM1 DE/DP




### 3.2-2 Wiring

Install the cable glands contained in the T-junction box and connect the bus as shown in the adjacent diagram.

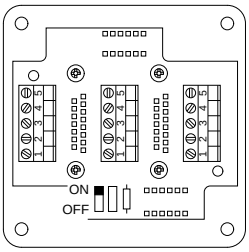
TSX CSA 100/200/500 cable




### 3.2-3 Line termination

When the TSX SCA 50 T-junction boxes are connected directly to a device at the end of a line, this line must be closed using the line termination circuit.

Set the jumper to the ON position, as shown (the products are supplied in the OFF position).



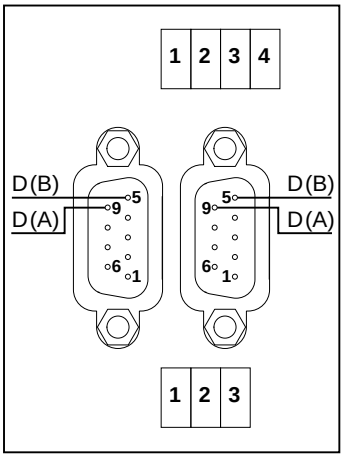
## 3.3 Installing the TBX 0010

Each T-junction box has two 9-pin Sub-D connectors for connecting two PLCs to the network, and two screw connectors for connection to the bus.
This T-junction box enables a Bus type network to be wired more conveniently than by creating tap links directly on the pins of the SUB-D connector.

This type of connection is used to link a network station without leaving the connector "pins open".

In addition, it makes it easier to connect new stations at a later date.

The junction also enables line end matching when it is located at the end of a network.




Key :
- D(A) in TSX 37 corresponds to L+ or Tx+ for S 1000,
- D(B) in TSX 37 corresponds to L- or Tx- for S 1000.

See reference manual TEM 60000E for wiring a network.

## 4.1 Details of Modbus/Jbus frames

### 4.1-1 Read n bits

**Read n output bits**

**Code : 01** : This function is used to access output bits or internal bits which can be read or written, and are defined in a Slave memory.

**Read n input bits**

**Code : 02** : This function, which is identical to the previous one and has the same limits, applies to input bits (bits which the Master can read but not write).

**Read n bits** : function 1 or 2

*Question :*

| Slave Number | 1 or 2 | N° of 1st bit | | Bit number | | CRC 16 |
|---|---|---|---|---|---|---|
| | | PF | Pf | PF | Pf | |
| 1 byte | 1 byte | 2 bytes | | 2 bytes | | 2 bytes |

*Response :*

| Slave Number | 1 or 2 | Number of bytes read | Value | ••••••••• | Value | CRC 16 |
|---|---|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | | | | 2 bytes |

Example : read bit %M3 of Slave 2

| | | | | | |
|---|---|---|---|---|---|
| **Question** | 02 | 01 | 0003 | 0001 | CRC 16 |
| **Response** | 02 | 01 | 01 | xx | CRC 16 |

xx = 00 if %M3 = 0

xx = 01 if %M3 = 1

### 4.1-2 Read n words

**Read n output words**

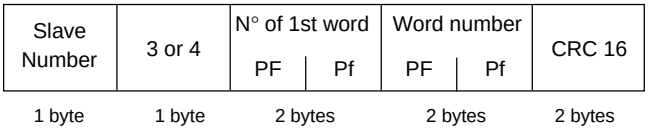
**Code : 03** : This function is used to read internal words or output words which can be read or written, and are defined in a Slave memory.

**Read n input words**

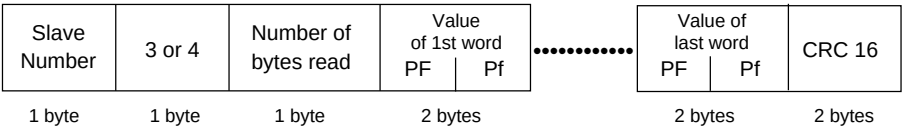
**Code : 04** : This function, which is identical to the previous one and has the same limits, applies to input words (words which the Master can read but not write).

**Read n words** : function 3 or 4

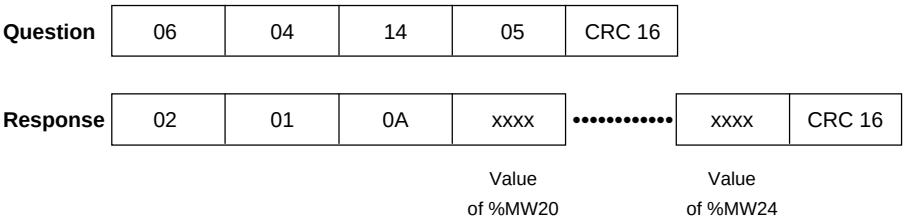
*Question :*




*Response :*




Example : read words %MW20 to %MW24 of Slave 6



### 4.1-3 Write output bit

**Code : 05** : This function is used to set an output bit which is defined in a slave memory to 0 or 1 (only accessible in write).

**Write output bit** : function 5

*Question :*

| Slave Number | 5 | Bit number PF \| Pf | Bit value | CRC 16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

The "Bit value" field has only two possible values :

- bit at 0 = 0000,
- bit at 1 = FF00.

*Response :*

| Slave Number | 5 | Bit number PF \| Pf | Bit value | CRC 16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

Example : write value 1 to bit %M3 of Slave 2

| | | | | | |
|---|---|---|---|---|---|
| **Question** | 02 | 05 | 03 | FF00 | CRC 16 |
| **Response** | 02 | 05 | 03 | FF00 | CRC 16 |

### 4.1-4 Write output word

**Code : 06** : This function writes a 16-output bit word defined in a Slave memory (only accessible in write).

**Write output word** : function 6

*Question :*

| Slave Number | 6 | Word number | | Word value | | CRC 16 |
|---|---|---|---|---|---|---|
| | | PF | Pf | PF | Pf | |
| 1 byte | 1 byte | 2 bytes | | 2 bytes | | 2 bytes |

*Response :*

| Slave Number | 6 | Word number | | Word value | | CRC 16 |
|---|---|---|---|---|---|---|
| | | PF | Pf | PF | Pf | |
| 1 byte | 1 byte | 2 bytes | | 2 bytes | | 2 bytes |

Example : write value H'3A15' to word %MW12 of Slave 5

| | | | | | |
|---|---|---|---|---|---|
| **Question** | 05 | 06 | 0C | 3A15 | CRC 16 |
| **Response** | 05 | 06 | 0C | 3A15 | CRC 16 |

### 4.1-5 Write n output bits

**Code : 15** : This function enables the Master to write output bits (which can be read or written) to a Slave memory.

**Write n output bits** : function 15 (H'0F')

*Question :*

| Slave Number | 0F | Address of 1st bit to force | Number of bits to force | Number of bytes | Value of bits to force | CRC 16 |
|---|---|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte | n bytes | 2 bytes |

*Response :*

| Slave Number | 0F | Address of 1st bit forced | Number of bits forced | CRC 16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

### 4.1-6 Write n output words

**Code : 16** : This function enables the Master to write output words (which can be read or written) to a Slave memory.

**Write n output words** : function 16 (H'10')

*Question :*

| Slave Number | 10 | Number of 1st word | Number of words | Number of bytes | Value of 1st word | ••••••••••• | CRC 16 |
|---|---|---|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte | 2 bytes | | 2 bytes |

*Response :*

| Slave Number | 10 | Number of 1st word | Number of words | CRC 16 |
|---|---|---|---|---|
| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes |

Example : Write values 1 and 2 to words %MW16 and %MW17 of Slave 11

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Question** | 0B | 10 | 0010 | 0002 | 04 | 0001 | 0002 | CRC 16 |
| **Response** | 0B | 10 | 0010 | 0002 | CRC 16 | | | |

## 4.2 Special characteristic of ASCII mode

**Structure of ASCII Modbus frame :**

| • • | Slave Number | Function code | Information | LRC PF \| Pf | CR | LF |
|---|---|---|---|---|---|---|
| 1 byte | 2 bytes | 2 bytes | n bytes | 2 bytes | 1 byte | 1 byte |

**LRC** : total content of the frame in hexadecimal, modulo FF, excluding delimiters, two's complement and coded in ASCII.

The delimiters are : (3A),..... CR (0D), LF (0A)

Example : command 8 (diagnostic) of Slave 1 in ASCII mode

| 3A | 30 31 | 30 38 | 30 30 30 30 36 31 36 32 | 33 34 | 0D 0A |
|---|---|---|---|---|---|
| Delimiter | Slave n° | Function code | Information | LRC | Delimiters |

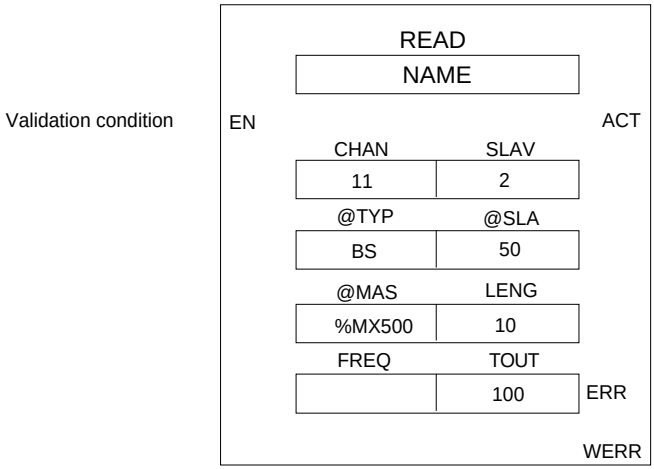
## 4.3 Examples

Examples of inter-PLC dialog via Jbus between an API 5000 and a TSX 37.

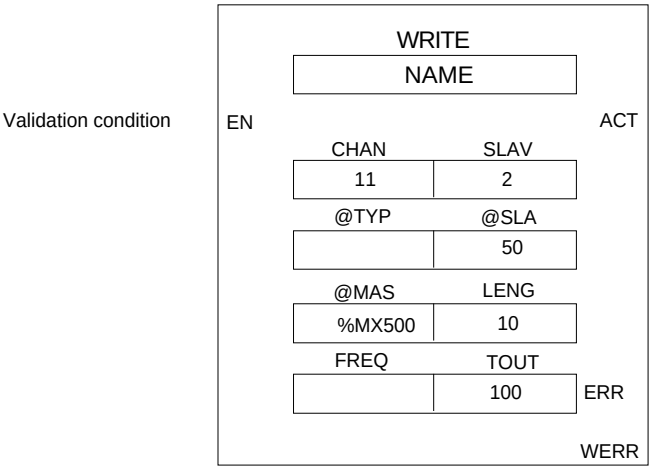
### 4.3-1 API 5000 master and TSX 37 slave

The API 5000 with a Jbus S0550 module in slot 1 channel 1 configured as master communicates with a TSX 37 equipped with a PCMCIA SCP114 configured as slave at address 2.

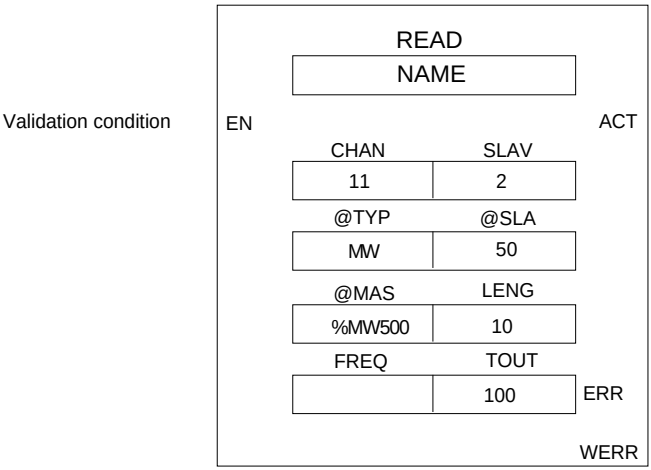
The objective is to read 10 bits in the TSX 37 at the addresses of internal bits %M50 to %M59, then save them in %MX500 to %MX509 in the API 5000 :



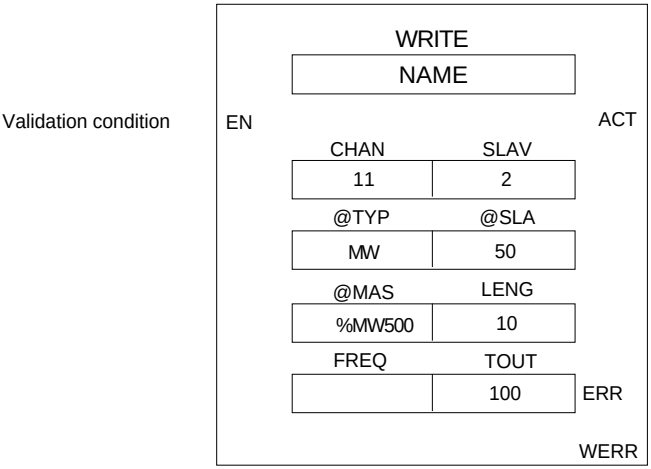

The objective is to write 10 bits to the TSX 37 at addresses %M50 to %M59, from %MX500 to %MX 509 in the API 5000 :



The objective is to read 10 words in the TSX 37 at the addresses of internal words %MW50 to %MW59, then save them in %MW500 to %MW509 in the API 5000 :




The objective is to write 10 words to the TSX 37 at addresses %MW50 to %MW59, from %MW500 to %MW 509 in the API 5000 :



### 4.3-2 TSX 37 master API 5000 slave

The API 5000 equipped with a Jbus S0550 module in slot 1 channel 1 configured as slave at address 2 communicates with a TSX 37 equipped with a SCP114 PCMCIA card configured as master.

**Read 10 bits** in the API 5000 at the addresses of internal bits %M500 to %M509, then save them in %MW100 in the TSX 37.

READ_VAR(ADR#0.1.2,'%M',%MD80,10,%MW100:1,%MW720:4)

with %MD80:= 16#0000A1F4
shift of 16#A000 + 16#1F4 (16#1F4=10#500)

**Write 10 bits** to the API 5000 at the addresses of internal bits %M500 to %M509, with the bits contained in %MW25:1 in the TSX 37.

. WRITE_VAR(ADR#0.1.2,'%M',%MD80,10,%M25:1,%MW720:4)

with %MD80:= 16#0000A1F4
shift of 16#A000 + 16#1F4 (16#1F4=10#500)

**Read 10 words** in the API 5000 at the addresses of internal words %MW500 to %MW509, then save them in %MW100 to %MW109 in the TSX 37.

READ_VAR(ADR#0.1.2,'%MW',%MD80,10,%MW100:10,%MW720:4)

with %MD80:= 16#000001F4 (16#1F4=10#500)

**Write 10 words** to the API 5000 at the addresses of internal words %MW500 to %MW509, from %MW100:10 in the TSX 37.

WRITE_VAR(ADR#0.1.2,'%M'W,%MD80,10,%MW100:10,%MW720:4)

with %MD80:= 16#000001F4 (16#1F4=10#500)

## Index