Premium, Micro PLCs

TSX CPP 100 / 110

07/2008 eng

35012328.02

www.schneider-electric.com

Table of Contents



About the Book 7

Chapter 1 General 9
At a Glance 9
Principles 10
General architecture of the CANopen field bus 11
Transmission speed and bus length 12

Chapter 2 Overview of the TSX CPP 100/110 PCMCIA cards 15
At a Glance 15
2.1 Description of the TSX CPP 100/110 card 16
At a Glance 16
Information Concerning TSX CPP 100 and TSX CPP 110 Cards 17
Mounting the TSX CPP 100 Card 19
Mounting the TSX CPP 110 Card 21
Connection unit TSX CPP ACC1 23
2.2 Technical specifications 25
At a Glance 25
Standards and Characteristics 26
CANopen Characteristics 27
Processors which support the TSX CPP 100/110 Card 28

Chapter 3 Software implementation 31
At a Glance 31
3.1 General 32
At a Glance 32
Installation Principle 33
Implementation Methodology 35
3.2 Configuration 37
At a Glance 37
How to Access the Configuration Screen 38
Configuration Screen 39
Description of the I/O Data and the Bus Behavior at the Start 41
Description of how to select configuration loading mode 44
How to load a configuration using X-Way 46

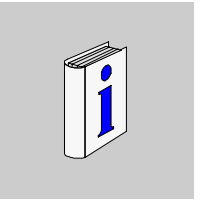
Slave configuration screen. 49
How to Configure a CANopen PCMCIA Card 50
Configuration Folder of the PCMCIA TSX CPP 100/110 Card 53
3.3 Programming 55
At a Glance 55
Access to CANopen SDOs 56
Accessing the CAN link layer PDUs 61
IDENTIFICATION Request 63
Diagnostic Command. 66
3.4 Debugging 68
Description of the Debug Screen. 68
3.5 Diagnostics 70
At a Glance 70
Diagnostics Using the Status LEDs on the TSX CPP 100/110 PCMCIA Card . . 71
Diagnostics data 72
How to perform a diagnostic 75
3.6 Language objects associated with the TSX CPP 100/110 card. 79
At a Glance 79
Implicit Exchange Language Objects. 80
Language objects for explicit exchange. 84
Explicit exchange management. 85
Language objects associated with configuration 86
Error codes 88

Chapter 4 Examples of Configuration with SyCon Tool 95
At a Glance 95
4.1 Standard configuration. 96
At a Glance 96
Example Description 97
Declaration of CANopen Master 98
CANopen Bus Configuration 100
Declaration of slave 7 101
Configuration of slave 7. 103
Declaration of slave 8 107
Configuration of slave 8. 108
Checking the configuration 109
4.2 Multi-master Configuration. 112
At a Glance 112
Example Description 113
Declaration of CANopen Master A and slaves 7 and 8 115
Declaration of Master B CANopen. 116
Master B CANopen Bus Configuration 118
Declaration and Configuration of Slaves 9 and 10. 119
Declaration and Configuration of Slave 7 for Master B 120
Declaration and Configuration of Slave 8 for Master B 125

Glossary **131**

Index **135**

About the Book



At a Glance

Document Scope

This manual is intended for users wishing to install the CANopen PCMCIA communication card:

- TSX CPP 100 on Premium,
- TSX CPP 110 on Premium or Micro.

Validity Note

This documentation describes the installation procedure for TSX CPP 100 or TSX CPP 110 PCMCIA cards using PL7 V4.5 and SyCon V2.8.

Related Documents

Title of Documentation	Reference Number
Common application-specific documentation for communication	TLX DSCOM PL7 xxE

Product Related Warnings

User Comments

We welcome your comments about this document. You can reach us by e-mail at techpub@schneider-electric.com

General

1

At a Glance

Aim of this Chapter

This chapter introduces the main features of CANopen communication.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Principles	10
General architecture of the CANopen field bus	11
Transmission speed and bus length	12

Principles

Introduction

Originally developed for automotive vehicle borne systems, the CAN communication bus is now used in many areas, such as:

- transport,
- moving part devices,
- medical devices,
- building,
- industrial control.

The advantages of the CAN system are:

- the bus allocation system,
- error detection,
- data exchange reliability.

Master/slave structure

The CAN bus has a master/slave structure for bus management.

The master manages

- slave initialization,
- communication errors,
- slave status.

Peer to peer communication

Communications on the bus are made **peer to peer**; at any moment, each device can send a request on the bus and the relevant devices reply. The priority of the requests circulating on the bus is determined by an identifier at individual message level.

CAN identifiers

Explicit exchanges of CAN PDUs at link level (see *Accessing the CAN link layer PDUs, p. 61*) use extended identifiers over 29 bits (CAN standard V2.0B).

11 bit identifiers (CAN standard V2.0A) can be used for sending, but the receiving of this type of identifier is not supported.

General architecture of the CANopen field bus

At a Glance

The CANopen architecture consists of:

- a bus master (TSX CPP 1•• PCMCIA card),
- slave devices.

Note: it is also possible to connect several TSX CPP 1•• PCMCIA cards on the bus, with one as master and all the others in **listen** mode. The cards in **listen** mode enable the Premium PLCs to which they are connected to know the status of the bus and the bus slaves at all times.

Note: the TSX CPP 100 PCMCIA card manages two bus segments (A and B) by means of the TSX CPP ACC1 connection unit.

The baud rate of the bus depends on its length and the cable type used (see *Transmission speed and bus length, p. 12*).

Transmission speed and bus length

At a Glance

CANopen allows 128 devices (the bus master and 127 remote slaves).
Transmission speed depends strictly on the bus length and the type of cable used.
The following two examples enable you to evaluate authorized values.

Example 1

The following table describes the relationship between the maximum transmission speed and the bus length in relation to the type of cable used, for a maximum of 32 slaves, without repeater, on segments A or B of the TSX CPP 100 card or on the single segment of the TSX CPP 110 card.

Transmission baud rate	Bus length	Cable characteristics
1 Mbit/s	25 m	0.25 mm^2, AWG 23
800 Kbit/s	50 m	0.34 mm^2, AWG 22
500 Kbit/s	100 m	0.34 mm^2, AWG 22
250 Kbit/s	250 m	0.34 mm^2, AWG 22
125 Kbit/s	500 m	0.5 mm^2, AWG 20
50 Kbit/s	1,000 m	0.75 mm^2, AWG 18
20 Kbit/s	1,000 m	0.75 mm^2, AWG 18
10 Kbit/s	1,000 m	0.75 mm^2, AWG 18

Example 2

The following table describes the relationship between the maximum transmission speed and the bus length in relation to the type of cable used, for a maximum of 100 slaves, without repeater, on segments A or B of the TSX CPP 100 card or on the single segment of the TSX CPP 110 card.

Transmission baud rate	Bus length	Cable characteristics
1 Mbit/s	25 m	0.34 mm^2, AWG 22
800 Kbit/s	50 m	0.6 mm^2, AWG 20
500 Kbit/s	100 m	0.6 mm^2, AWG 20
250 Kbit/s	250 m	0.6 mm^2, AWG 20
125 Kbit/s	500 m	0.75 mm^2, AWG 18
50 Kbit/s	1,000 m	1 mm^2, AWG 17
20 Kbit/s	1,000 m	1 mm^2, AWG 17
10 Kbit/s	1,000 m	1 mm^2, AWG 17

Overview of the TSX CPP 100/110 PCMCIA cards

2

At a Glance

Aim of this Chapter

This chapter describes the main technical characteristics of TSX CPP 100/110 PCMCIA cards.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
2.1	Description of the TSX CPP 100/110 card	16
2.2	Technical specifications	25

2.1 Description of the TSX CPP 100/110 card

At a Glance

Subject of this Section

This section describes the physical characteristics of the PCMCIA TSX CPP 1•• card and its connections.

What's in this Section?

This section contains the following topics:

Topic	Page
Information Concerning TSX CPP 100 and TSX CPP 110 Cards	17
Mounting the TSX CPP 100 Card	19
Mounting the TSX CPP 110 Card	21
Connection unit TSX CPP ACC1	23

Information Concerning TSX CPP 100 and TSX CPP 110 Cards

At a Glance

The CANopen communication card TSX CPP 1•• is used to implement a CANopen architecture. This card is master of the bus and enables the connection of devices in compliance with the CANopen standard:

- Implicit exchange of **P**rocess **D**ata **O**bjects using %MW words.
- Explicit exchange of **S**ervice **D**ata **O**bjects via READ_VAR and WRITE_VAR function blocks.
- Compatibility with standardized device profiles and communication on CANopen (2.0A and 2.0B).

Physical Description

The TSX CPP 1•• card is a type III PCMCIA card which is inserted into the PCMCIA communication slot on the processor.

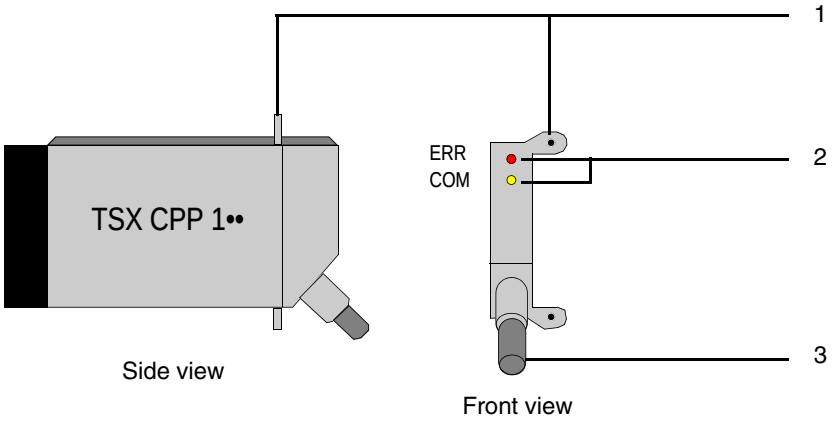
Type III CANopen PC cards (TSX CPP 1••) can be used in the PCMCIA slots of the following types of Premium CPUs:

- TSX P57103, TSX P572•3, TSX P572•23, TSX P573•3, TSX P57453, TSX P573623, TSX P574823, T PCX 57203, T PCX 57353.

Type III CANopen PC cards (TSX CPP 110) can be used in the PCMCIA slots of the following types of Micro CPUs:

- TSX 3721•••1

This module is made up of the following elements:



Description

This table describes the elements in the previous figure.

Address	Description
1	Fixing brackets, on the top and bottom of the card, which are used to attach it to the processor.
2	LEDs, which are used for diagnostics of the working of the communication card (see *Diagnostics, p. 71*).
3	• TSX CPP 100: bus cable. This 60 cm cable has a male 15 pin Sub-D connector at the end to connect to the TSX CPP ACC1. (see *Connection unit TSX CPP ACC1, p. 23*) • TSX CPP 110: bus cable, this 50 cm cable is equipped at its end with a CANopen industrial TAP.

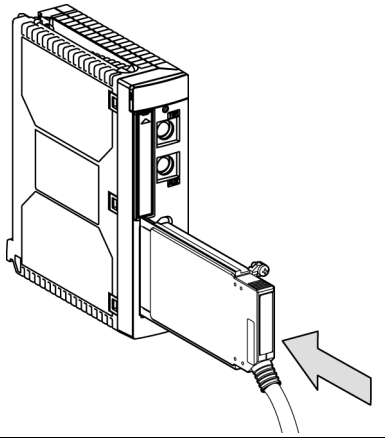
Mounting the TSX CPP 100 Card

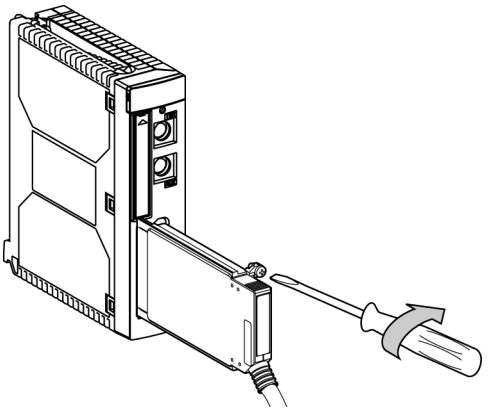
At a Glance

This card is inserted in the slot reserved for communication cards. The bus connection cable is supplied with the card and is connected to the TSX CPP ACC1 device (also called TAP) using the Sub-D 15-pin connector.

	CAUTION
	To insert or disengage a PCMCIA card the PLC must be switched off. **Failure to follow these instructions can result in injury or equipment damage.**

The following table describes the procedure for installing a TSX CPP 100 card.

Step	Action	Illustration for a Premium PLC
1	Switch off the PLC.	
2	Insert the CANopen type III PC card in the PCMCIA slot of the Premium CPU.	

Step	Action	Illustration for a Premium PLC
3	Screw the card in securely to ensure it operates correctly.	
4	Connect the card's cable to the TSX CPP ACC1 device using the 15-pin Sub-D connector, remembering to tighten the fixing screws, then fix the TAP to a DIN rail.	
5	Switch the PLC on again.	

Mounting the TSX CPP 110 Card

PCMCIA card TSX CPP 110

The PC TSX CPP 110 card with its industrial TAP operate the link between the Premium or Micro CPU and a CANopen network.

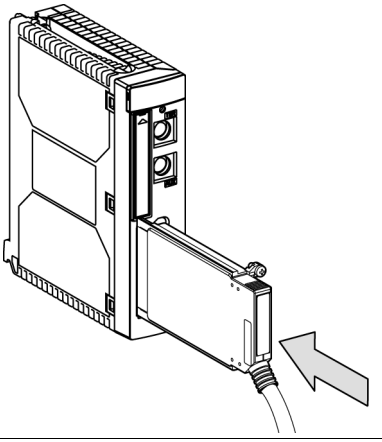
Note: the Modbus TSX SCY 2160• communication module cannot be used.

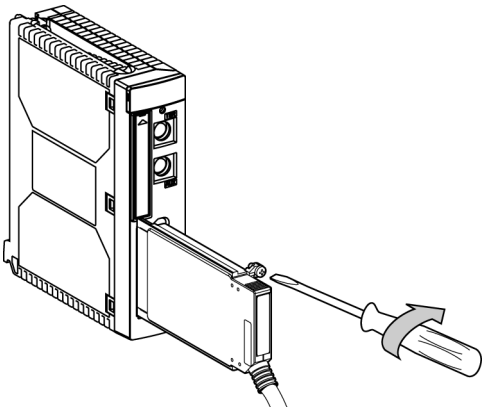
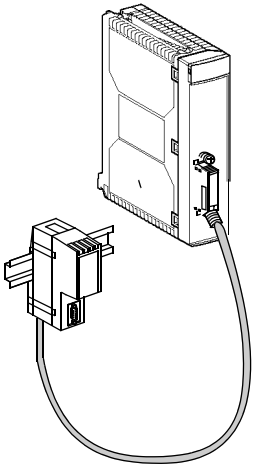
CAUTION

To insert or disengage a PCMCIA card the PLC must be switched off.

Failure to follow these instructions can result in injury or equipment damage.

The following table describes the procedure for installing a TSX CPP 110 card.

Step	Action	Illustration for a Premium PLC
1	Switch off the PLC.	
2	Insert the CANopen type III PC card in the PCMCIA slot of the Premium CPU.	

Step	Action	Illustration for a Premium PLC
3	Screw the card in securely to ensure it operates correctly.	
4	Fix the TAP on a DIN rail.	
5	Switch the PLC on again.	

Connection unit TSX CPP ACC1

At a Glance

The CANopen TSX CPP 100 card connects to 15 female pin Sub-D connector in the TSX CPP ACC1 connection unit with the cable supplied with the card. This unit allows you to connect two CANopen bus segments (it allows you to double the total length of the bus) using the two 9-pin Sub-D connectors.

The cables and other connection accessories are available from specialized companies.

TSX CPP ACC1 connection unit

This figure illustrates a TSX CPP ACC1 connection unit.

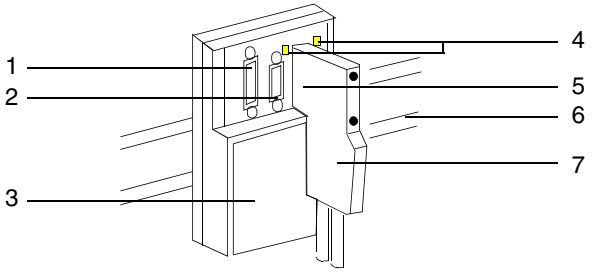


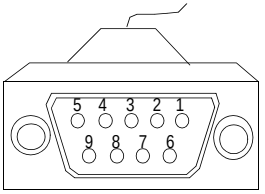
Figure description

This table describes the elements in the previous figure.

Element	Description
1	**15-pin Sub-D connector**: the TSX CPP 100 card connects using this connector.
2	**9-pin male Sub-D connector**: connector **A** for connecting the **A** CANopen segment.
3	TSX CPP ACC1 connection unit.
4	Display LEDs for the bus operation via the TSX CPP ACC1 unit (Off = no communication via CPP 100).
5	**9-pin male Sub-D connector**: connector **B** for connecting the **B** CANopen segment.
6	Fixed top hat rail.
7	CANopen bus connector , with or without line terminal.

9-pin Sub-D connector

This figure illustrates bus connector A or B, 9-pin female Sub-D connector of a bus segment on the TSX CPP ACC1 connection unit.



Connector description

This table describes the connection.

Element	Description
1	Reserved
2	CAN_L
3	CAN_GND
4	Reserved
5	NC
6	NC
7	CAN_H
8	Reserved
9	NC

Note: this connector pin assignment is also valid for the CANopen interface of the TSX CPP 110 tap.

2.2 Technical specifications

At a Glance

Subject of this Section

This section describes the main technical characteristics of the TSX CPP 1•• card.

What's in this Section?

This section contains the following topics:

Topic	Page
Standards and Characteristics	26
CANopen Characteristics	27
Processors which support the TSX CPP 100/110 Card	28

Standards and Characteristics

Standards

The TSX CPP 1•• communication card complies with the following international standards:

International Standards	ISO IS 11898, CAN High Speed Transceiver and Data Link Layer
US Standards	UL508
Radiation Standards	EC label, FCC-B (50082-1)

Electrical Characteristics

- Logical V DC supply: 5 V provided by the rack supply
- Power Consumption: 3W

Environment Characteristics

- Storage Temperature: -25 °C to 70 °C
- Operating Temperature: 0° C to 70 °C
- Storage Hydrometry: 30% to 95% without condensation
- Operating Hydrometry: 5 % to 95 % without condensation.

CANopen Characteristics

Standards

The TSX CPP100 communication card is compliant with the standard DS301 V3.0.

The TSX CPP110 communication card is compliant with the standard DS301 V4.01.

Specific features

- the user may select a PDO content mapping compliant with the standard DS301 V4.01,
- the TSX CPP 110 card supports the "heartbeat" function (DS 301 V4.01)
- the TSX CPP 1•• card is normally network management master (NMT_MASTER) on the bus (this function can be disabled via SyCon),
- the TSX CPP 1•• card normally produces the synchronization (SYNC) variable (this function can be disabled via SyCon),
- the Node ID of card TSX CPP 1•• may not use for data transfer. It is only used for the "heartbeat" function with card TSX CPP 110.

Processors which support the TSX CPP 100/110 Card

At a Glance

- CPU version:
  - TSX CPP 100 and TSX CPP 110 cards can only be used on Premium and Atrium CPU versions ≥5.0.
  - TSX CPP 110 cards can only be used on Micro CPU versions ≥ 6.0.

TSX CPP 110 cards are implemented using PL7 V4.4 software.

The general configuration of a CANopen bus is defined using **SyCon** software version 2.630 or later (TLX LFBCM), TSX CPP 110 card version 2.8 or later.

Note: it is mandatory for the PCMCIA card to be installed in the slot located in the processor module. As a result, only one CANopen bus is available for each PLC CPU.

Types of Processors and Capacities

The following table describes the processors that support the CANopen PCMCIA card, TSX CPP 100/110, and their maximum storage capacities.

Processor	**Maximum size of TSX CPP 100/110 configuration data located in the processor**	**Maximum size of input/output data for the CANopen node configuration**	
		MAST task	FAST task
TSX 372•••1	0 to 8 Kb	256 %MW (128+128)	32 %MW (16+16)
TSX P57103	0 to 12 Kb	384 %MW (192+192)	48 %MW (24+24)
TSX P57203 T PCX 57 203 TSX P572623 TSX P57253 TSX P572823	16 Kb	512 %MW (256+256)	64 %MW (32+32)
TSX P57303 TSX P57353 TSX P573623 T PCX 57353	32 Kb	1024 %MW (512+512)	256 %MW (128+128)
TSX P57453 TSX P574823	64 Kb	3584 %MW (1792+1792)	512 %MW (256+256)
Legend	(1): This maximum size can be exceeded if you load the configuration into the card using the SyCon software (see *Description of how to select configuration loading mode, p. 44*). The maximum size of configuration data authorized by the SyCon software is 128 Kb.		

Note: The effective size of the configuration is given in the word %KWy.1.2 (see *Language objects associated with configuration, p. 86*).

Software implementation

3

At a Glance

Aim of this Chapter

This chapter describes the various configuration, operation and diagnostics options for a CANopen application.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
3.1	General	32
3.2	Configuration	37
3.3	Programming	55
3.4	Debugging	68
3.5	Diagnostics	70
3.6	Language objects associated with the TSX CPP 100/110 card	79

3.1 General

At a Glance

Subject of this Section

This section describes the software installation of a TSX CPP 1•• PCMCIA card.

What's in this Section?

This section contains the following topics:

Topic	Page
Installation Principle	33
Implementation Methodology	35

Installation Principle

At a Glance

In order to install a CANopen bus, it is necessary to define the physical context of the application in which it is to be integrated (rack, supply, processor, modules or devices, etc.) then ensure the necessary software is installed.

The software is installed using different PL7 editors:

- Either in offline mode,
- Or in online mode: in this case, modification is limited to certain parameters.

The bus is configured using SyCon software.

Installation Principle

The following table shows the different installation phases.

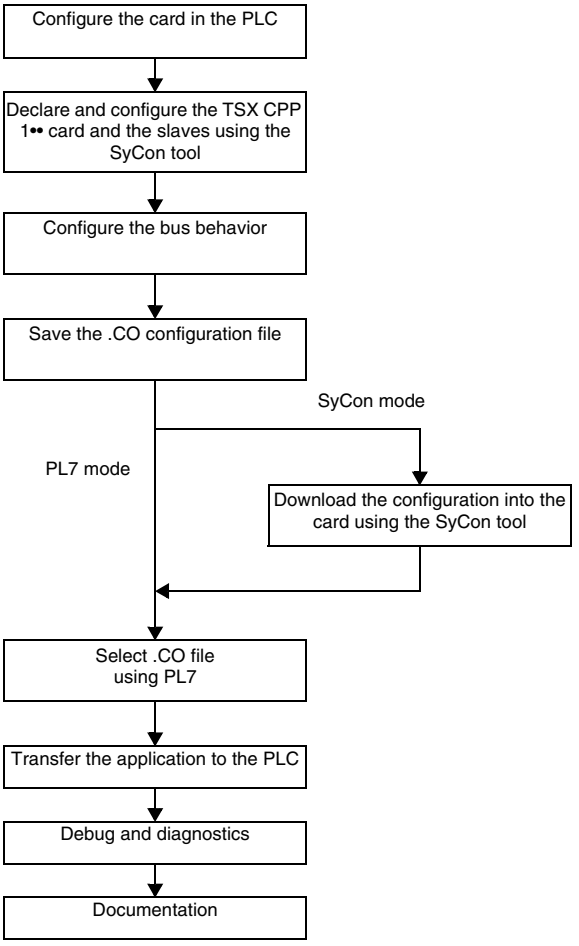
Mode	Phase	Description
Local	Declaration of the TSX CPP 100/110 PCMCIA card	The card must be installed in the type III PCMCIA slot of the processor.
	Configuration	• Entering the configuration parameters, • Declaring the bus configuration using **SyCon** software and generating of the *.CO configuration file, • Selecting the *.CO configuration file using PL7.
Offline or online	Symbolization	Symbolization of the variables associated with the CANopen card.
	Programming	Programming the specific functions: • Bit objects and associated words, • Specific instructions.
Online	Transfer	Transferring the application to the PLC Transferring the application to the PLC, or a cold start of the application starts and configures the TSX CPP 1•• card.
	Debug Diagnostics	Various resources are available for debugging the application, controlling inputs/outputs and diagnosing faults: • PL7 language objects, • The PL7 debug screen, • LED signaling.
Offline or online	Documentation	Printing the various data relating to the configuration of the TSX CPP 1•• card.

Note: the order shown above is provided for information. With PL7 software, editors can be used in the order you wish, and interactively (however, the data or program editor cannot be used without having first performed the configuration).

Implementation Methodology

Overview

The following flowchart shows the implementation methodology for a TSX CPP 1•• card.



Recommendations

When configuration of the CANopen bus is too great to switch to PL7 mode, the following checks are recommended:

- See if you can remain in PL7 mode using a more powerful processor (see *Types of Processors and Capacities, p. 29*),
- Change to Sycon mode, if it is not possible to select a more powerful processor.

3.2 Configuration

At a Glance

Subject of this Section

This section describes the configuration of a TSX CPP 1•• PCMCIA card.

What's in this Section?

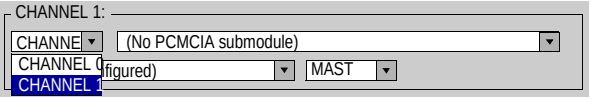
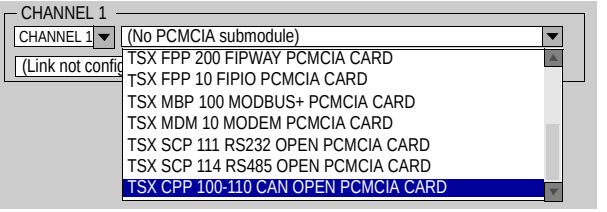
This section contains the following topics:

Topic	Page
How to Access the Configuration Screen	38
Configuration Screen	39
Description of the I/O Data and the Bus Behavior at the Start	41
Description of how to select configuration loading mode	44
How to load a configuration using X-Way	46
Slave configuration screen	49
How to Configure a CANopen PCMCIA Card	50
Configuration Folder of the PCMCIA TSX CPP 100/110 Card	53

How to Access the Configuration Screen

Procedure

This operation is used to declare a TSX CPP 100 or TSX CPP 110 card in the PCMCIA type III slot of the processor. The example below describes the steps to be followed.

Step	Action
1	Open the hardware configuration editor using the application browser.
2	Access the configuration screen of the processor's communication channel by double-clicking the **Comm** zone of the processor.
3	Select the communication channel from the **CHANNEL 1** drop-down menu: **Example** 
4	From the drop-down menu, select the TSX CPP 100-110 PCMCIA card: **Example**  **Result**: a configuration screen for the CANopen bus appears (see *Configuration Screen, p. 39*).

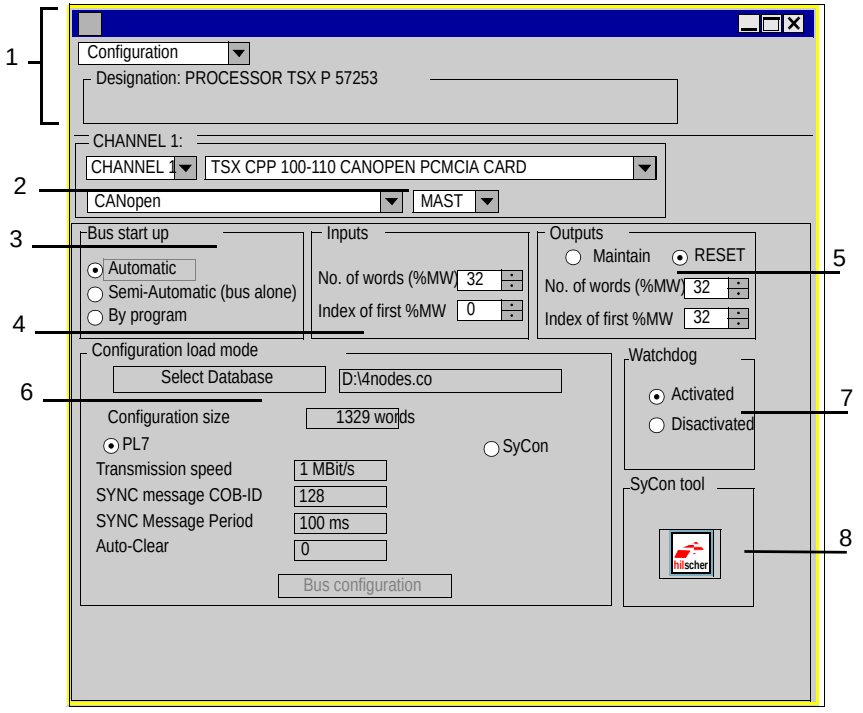
Configuration Screen

At a Glance

This screen is used to declare the communication channel and configure the parameters of a CANopen link.

Illustration

The screen dedicated to CANopen communication is as follows:



Elements and Functions

This table describes the various areas that make up the configuration screen:

Area	Number	Function
common	1	This area is made up of: ● A title bar stating the commercial reference and position of the module, ● A drop-down list used to select **configuration** or **debug** mode (in online mode only), ● A window displaying the name of the selected module.
specific	2	This drop-down list is made up of two options which define the update rate for the memory areas associated with the I/Os: ● MAST: update rate for the MAST task ● FAST: update rate for the FAST task
	3	This area is used to select the behavior of the bus on start-up.
	4	This area is used to configure the address (internal PLC memory) into which the inputs of the CANopen device will be periodically copied.
	5	This area is used to configure the fallback mode of the outputs of the bus devices, and the address (internal PLC memory) where the outputs of the CANopen device will be periodically read.
	6	This area is used to configure the bus: ● Selection of the SyCon configuration file (*.CO) (see *How to Select a Configuration File, p. 50*) ● Configuration of PL7 or SyCon (see *Description of how to select configuration loading mode, p. 44*)
	7	This area is used to activate or deactivate the CANopen bus watchdog. The watchdog is activated by default. It is triggered when the PCMCIA card can no longer correctly manage the bus. When it is triggered, it switches the slave outputs to zero.
	8	This button is used to run the **SyCon** software, if installed on the PC.



WARNING

we recommend you pay close attention to the behavior of the bus when the watchdog is deactivated.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

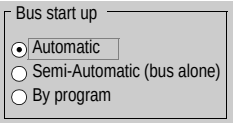
Description of the I/O Data and the Bus Behavior at the Start

At a Glance

The configuration screen allows you to configure the bus behavior at start-up as well as the inputs and outputs of slave devices on the bus.

Bus start-up

This figure illustrates the configuration zone of the bus start-up.



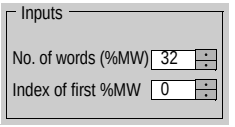
The bus start-up can be performed in three ways:

- **Automatic**: the bus configuration, communication control and the updating of E/S slaves are activated at start-up, without application intervention.
- **Semi-automatic**: the bus configuration and the communication control are activated at start-up but the E/S control should be confirmed by the application with the help of corresponding language objects (see *Output Word Objects, p. 83*).
- **By program (bus and I/O)**: The bus start-up should be fully managed by the application using the appropriate language objects (see *Output Word Objects, p. 83*).

Note: in automatic mode, if a bus fault appears and disappears, the bus automatically restarts the exchanges. However the fault information is latched for the fault bits (%MWy.1.2:x0 and %Iy.MOD.ERR set to 1). Acknowledgement is made by setting bit %QWy.1.0:x3=1 using the application or the animation table.

Inputs

This figure illustrates the input configuration zone.



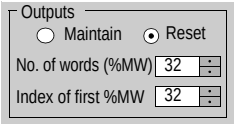
To configure slave bus inputs, it is necessary to indicate a memory zone in which they will be recopied periodically. To define this zone, indicate:

- **A number of words**: this is the number of input words corresponding to the size of input data configured by the Sycon**software**.
- **The address of the first word**: this is the address of the first word in the input memory zone.

Note: the %MW contain slave bus input values. In "PL7" load mode, it is possible to display words linked to slaves by clicking on the **Bus configuration** button. In the "Sycon" load mode, this button is greyed. The %MW linked to the slaves may be acquired in the following manner: First index marker + input/output address of the slave (see SyCon Software in the screen **View** → **Address Table ...**).

Outputs

This figure illustrates the output configuration zone.



To configure the outputs, it is necessary to indicate, as for the inputs, the word table that will contain the bus output value, but also the type of fallback needed when there is a slave fault passage:

- Maintain,
- Reset (RAZ)

Note: these %MW words contain output values of the slaves on the bus. In "PL7" load mode, it is possible to display words linked to slaves by clicking on the **Bus configuration** button. In the "Sycon" load mode, this button is greyed. The %MW linked to the slaves may be acquired in the following manner: First index marker + input/output address of the slave (see SyCon Software in the screen **View** → **Address Table ...**).

Note: the word tables are found in the PLC internal memory. The two zones should not overlap, as there is a risk of application malfunctioning.

Note: if the number of input or output words is different from what is determined in the configuration file (file name *.CO), the PL7 software signals it when confirming the configuration.

Note: the maximum authorized size of the memory zone reserved for E/S is a function of the processor type and the associated task (see *Types of Processors and Capacities, p. 29*).

Note: as for Fipio bus, if the %SW9 word is used (Set to 1 from %SW9:X0 bit for the master task and to 1 from %SW9:X1 for the high speed job), the bus output device remains as it was before being set to bit 1.

Description of how to select configuration loading mode

At a Glance

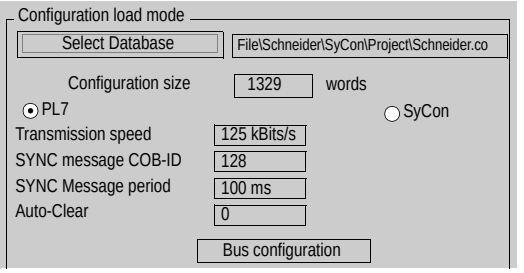
The configuration screen can be used to select the configuration loading mode. The two possible scenarios are as follows:

- loading by the PL7 software,
- loading by the Sycon software.

For each scenario, it is vital to select the configuration database (see *How to Select a Configuration File, p. 50*) created with the Sycon software.

Illustration

The following figure shows the area of the configuration screen used to select the configuration loading mode.



Description

The following table shows the different possible choices.

Zone	Description
Select Database	This area is used to select the database corresponding to the configuration of the bus managed by the PCMCIA TSX CPP 1•• card. **Note**: The selected database (file) must remain in the specified location, otherwise the next time the configuration screen is opened, an error message will warn you that the file could not be found. Access to the bus configuration screen will therefore not be possible. This configuration is performed using the Sycon software that generates the *.CO file which must be selected (see *How to Select a Configuration File, p. 50*).
PL7	When this button is selected, the bus configuration is loaded with the PL7 application. When the application is too large (memory size greater than that authorized for the processor), the PL7 software does not authorize this choice, and you must select **Sycon**, described below.
Sycon	When you select this button, the configuration of the bus is considered to have been already loaded in the PCMCIA card using the PC and Sycon software. The PL7 software simply performs a check in order to make sure the configuration of the card is identical to that of the *.CO file selected. Other than that, it is impossible to modify the parameters of the bus using PL7. To modify the configuration, you have to use the **Sycon** software.
Transmission speed	When the PL7 loading mode has been selected, this area displays the transmission speed on the bus defined in Sycon.
SYNC message COB-ID	When the PL7 loading mode has been selected, this area displays the SYNC Message COB-ID selected in Sycon.
SYNC message period	When the PL7 loading mode has been selected, this area displays the period of the bus defined in Sycon.
Auto-Clear	When the PL7 loading mode has been selected, this area displays the **Auto-clear on** or **Auto-clear off** mode selected in Sycon.
Bus configuration	When the PL7 loading mode has been selected, this button can be used to access the configuration of the slaves on the bus.

How to load a configuration using X-Way

General

When the CANopen bus is configured in load mode by Sycon, you can load the configuration onto the TSX CPP 1•• card using the X-Way Driver.

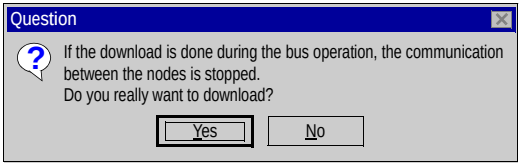
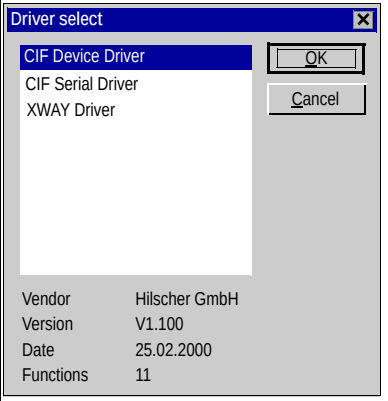
This loading can be carried out with an Ethernet network, or simply on a Unitelway bus.

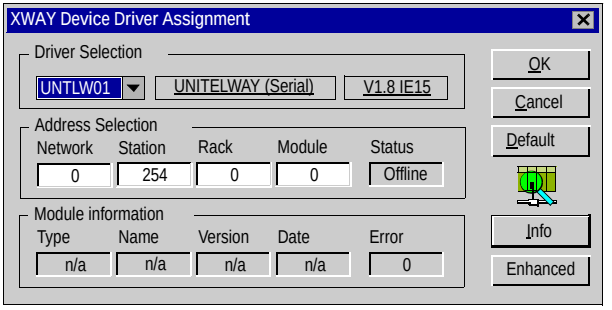
Note: MAKE SURE you always put the PLC in STOP mode during the downloading process.

Note: when the choice of X-Way communication driver has been validated, you must exit the Sycon software to change the driver.

Procedure

This table describes the steps to be carried out to load the configuration of a CANopen card using the X-Way communication drivers.

Step	Action
1	Connect to the PLC containing the TSX CPP 1•• card using the PL7 V4 software.
2	Switch this PLC to STOP mode.
3	Launch the Sycon software.
4	Load or create the desired configuration using the Sycon software.
5	Select the command **Online** → **Download**. **Result**: a message appears, which indicates that during configuration loading, communication between the slaves will be stopped. 
6	Click **YES** to indicate that you accept this halt in inter-slave communication. **Result**: a selection window for the **X-Way** or **CIF** driver appears. 

Step	Action
7	Select the **X-Way** driver then click **OK**. **Result**: the **Assigning the X-Way driver** window appears. 
8	Select the required driver (Uni-telway, XIP, etc.) from the **Driver selection** zone.
9	Enter the PLC address (Network, Station, Module, Rack) then click **OK** to start the loading process. **Result**: during loading, a window indicates the progress of the data transfer. When the transfer is complete, this window disappears and makes way for the main screen of bus configuration.

SyCon command "Online → Firmware / Reset"

In the same way as for the SyCon command "**Online → Download**", the online command "**Reset**" leads to all slaves on the bus being disconnected.

Note: once it receives the "Reset" command from Sycon, the TSX CPP 1•• card will not restart simply by switching the PLC to RUN mode: you therefore need to reset the PLC (RESET button),or switch the PLC off and on again, in order for the PCMCIA card to function.

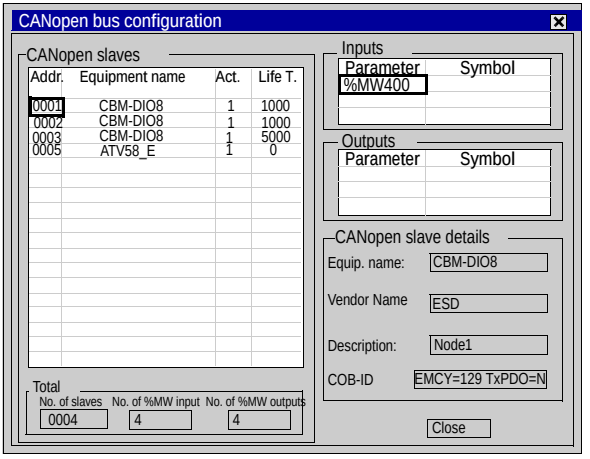
Slave configuration screen

At a Glance

With PL7 software, it is possible to access bus slave configuration. The information on the screen is almost identical to the debugging screen information (see *Description of the Debug Screen, p. 68*).

Illustration

The following diagram describes the screen which displays slave configuration.



Operation

- Click on a slave from the list of CANopen slaves.
- The slave parameters in inputs and outputs appear in the **Inputs** and **Outputs** zones.
- Slave information appears in the **CANopen device details** zone.

How to Configure a CANopen PCMCIA Card

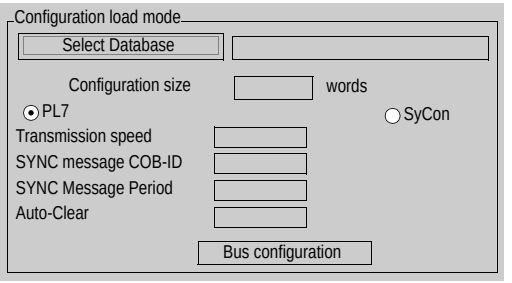
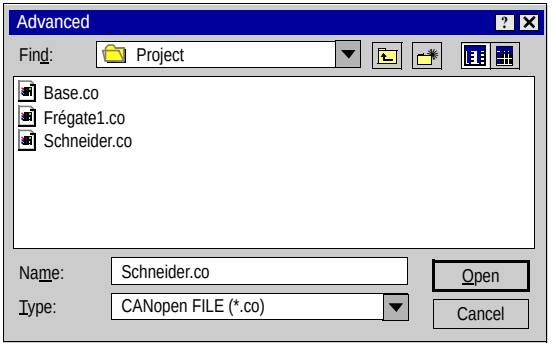
At a Glance

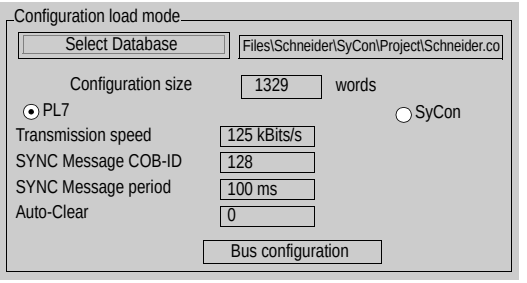
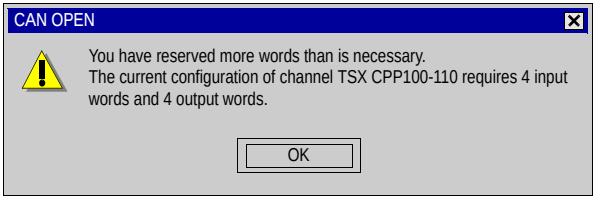
To configure a TSX CPP 1•• card, certain procedures are essential or require special explanation. Details of these procedures are given below.

Note: once the *.CO configuration file is selected (see following procedure), it must always be present. Otherwise the next time the configuration screen is opened, an error message will appear warning that the configured file can not be found. Access to the bus configuration screen will not be possible.

How to Select a Configuration File

This table describes the steps in the procedure for selecting a CANopen configuration.

Step	Action
1	Click **Select Database**:  **Result**: a screen like this appears: 

Step	Action
2	Select the desired *.CO file then click on **Open**. **Result**: if the number of words reserved for inputs and outputs corresponds to the selected configuration, the configuration appears in the PL7 configuration screen.  Otherwise, a warning message informs you that the number of input or output words does not correspond to the file selected. **Example**:  When the .CO file exceeds the maximum capacity for configuration data (see *Types of Processors and Capacities, p. 29*), an error message appears. You must then either use a more powerful Premium processor, or switch to **SyCon** mode and load the configuration directly into the card using the **SyCon** software.

Procedure for Configuring a CANopen Card

The following table describes the procedure for configuring a CANopen TSX CPP 1•• PCMCIA card.

Step	Action
1	Select the type of bus start-up
2	Click **SyCon Tool** to run the **SyCon** configuration software.
3	Using the **SyCon** software, configure your CANopen bus according to the devices planned for the bus.
4	Note the size of the buffers to be reserved for inputs and outputs for configuration of the PLC tables.
5	Save the configuration in a file with the .CO extension.
6	Return to PL7.
7	Enter the number of input and output words, indicating the values that you noted at Step 4. **Note**: Warning, each %MW is made up of two bytes.
8	Enter the start addresses for the tables, so there is no overlap between inputs and outputs.
9	Select a configuration file (see *How to Select a Configuration File, p. 50*).
10	Select the PL7 radio button if you want to incorporate the CANopen configuration into the PL7 application.
11	Click on the Watchdog **Activated** button.
12	Confirm the configuration.

Configuration Folder of the PCMCIA TSX CPP 100/110 Card

At a Glance

Using the Processor module, a folder is available in the PL7 documentation editor which describes how to configure the application for a PCMCIA TSX CPP 1•• card.

Illustration

The folder is presented as follows:

	TSX 57353 [RACK 0 POSITION 0]		
Module identification			
Commercial ref.:	TSX57353	**Designation:**	PROCESSOR TSX P 57353
Address:	000	**Symbol:**	
Channel parameters: 0			
Task/channel Assignment:	MAST		
Channel type:	Terminal port	Channel symbol:	
Application-specific function:	UNI-TELWAY LINK	Channel symbol:	
Transmission rate:	19200 Bits/s	Delay:	30 ms
Type of module:	Master	Parity:	Odd
Number of slaves:	8		
Channel Parameters: 1			
Task/channel Assignment:	MAST		
Type of sub-module:	TSX CPP 100-110 CANOPEN PCMCIA CARD		
Channel type:	PCMCIA port	Channel symbol:	
Application-specific function:	CANOPEN		
Inputs	Address of the first %MW	32	Length: 424
Outputs	Address of the first %MW	1056	Length: 102
	Output fallback strategy: Reset to zero		
Configuration mode:	Automatic	TSX CPP 100-110 watchdog:	active
Load mode,	PL7	CANOPEN configuration file:	E:\DB1.CO

Config. of the CANopen bus:	Transmission speed:	1 MBits/s	Sync message COB-ID:	128
	Auto-Clear:	off	Sync Message Period:	100 ms

CANOPEN slave configuration

Addr.	Type	Act./Guard poll
1	ATV_58F	1 / 1
2	Profile 401 standard EDS	1 / 0
3	CBM-DIO8	1 / 0

CANOPEN slave language objects:

Addr.	Inputs	Symbol	Outputs	Symbol
1.	%MW32		%MW1056	
	%MW33			
	%MW34			
	%MW35		%MW1057	
	%MW36		%MW1058	

3.3 Programming

At a Glance

Subject of this Section

This section describes the tools available to program the operation of and obtain information on a CANopen bus managed by the TSX CPP 1•• PCMCIA card.

It is possible to program the operation of the CANopen bus using UNI-TE requests:

- Send and receive SDO messages on the bus,
- Access the link layer by sending PDUs.

It is also possible to monitor the bus and its operation:

- Identification of the master,
- Send diagnostics requests on bus devices.

These requests are sent to the CANopen master (TSX CPP 1•• PCMCIA card) to be processed.

What's in this Section?

This section contains the following topics:

Topic	Page
Access to CANopen SDOs	56
Accessing the CAN link layer PDUs	61
IDENTIFICATION Request	63
Diagnostic Command	66

Access to CANopen SDOs

Overview

The READ_VAR and WRITE_VAR communication functions facilitate the access to transfer data of type SDO CANopen. The parameters of these functions determine the action executed.

These services are based on the CMS normalized message service of the CANopen standard. Consult the documentation on the CANopen bonds to recognize the format of the SDOs used.

Syntax of the WRITE_VAR function

The syntax of the communication function is presented under the following form :

```
WRITE_VAR(ADR#y.1.SYS, 'SDO', subindex:index, NodeID, %MWi:L,
%MWk:4)
```

The following table describes the different parameters of the function.

Parameter	Description
ADR#y.1.SYS	Address of the destination entity of the exchange. ● y : processor slot in the rack (0 or 1) ● 1 : path (always 1) ● SYS : UNI-TE server on the PCMCIA card
'SDO'	SDO object type (always **SDO** in capitals)
subindex:index	Double word or immediate value identifying the index and the sub-index SDO CANopen : The high-order word consisting of the double word contains the sub-index of the index and the low-order word consists of the index
NodeID	Word or value identifying the destination equipment on the CANopen bus
%MWi:L	Word table containing data to send (minimum length = 1)
%MWk:4	Control parameters of the exchange. four words identifying the address of the PL7 data used to drive the WRITE_VAR function

Syntax of the READ_VAR function

The syntax of the communication function is presented under the following form :

```
READ_VAR(ADR#y.1.SYS, 'SDO', subindex:index, NodeID, %MWi:L,
%MWk:4)
```

The following table describes the different parameters of the function.

Parameter	Description
`ADR#y.1.SYS`	Address of the destination entity of the exchange. ● y : processor slot in the rack (0 or 1) ● 1 : path (always 1) ● SYS : UNI-TE server on the PCMCIA card
`'SDO'`	SDO object type (always **SDO** in capitals)
`subindex:index`	Double word or immediate value identifying the index and the sub-index SDO CANopen : The high-order word consisting of the double word contains the sub-index of the index and the low-order word consists of the index
`NodeID`	Word or value identifying the destination equipment on the CANopen bus
`%MWi:L`	Word table containing data to be received (minimum length = 1)
`%MWk:4`	Control parameters of the exchange. four words identifying the address of the PL7 data used to drive the `READ_VAR` function

Control parameters : The following table details the %MWk:4 words.

Word number	High-order octet	Low-order octet	Data managed by
%MWk	Exchange number	Activity bit	System
%MWk+1	Operating account value : Covers all errors in the CANopen services 16#00: Positive result 16#01: Request not handled 16#02: Incorrect response 16#03: Reserved	Communication account value : Covers all errors for the 'READ_VAR', 'WRITE_VAR', 'SEND_REQ' function blocks 16#00: Correct exchange	
	16#00 See the following 'control parameter example'.	16#01: Cease exchange at timeout 16#02: Cease exchange when requested by the user (CANCEL) 16#03: Incorrect address format 16#04: Incorrect destination address 16#05: Incorrect control parameter format 16#06: Incorrect specific parameters 16#07: Transmission problems to the destination 16#09: Buffer reception size is insufficient 16#08: Reserved 16#0A: Buffer transmission size is insufficient 16#0B: Lack of processor system resources 16#0C: Incorrect exchange number 16#0D: No telegram received 16#0E: Incorrect length 16#0F: Unconfigurated telegram service 16#10: Coupling network is absent 16#11: Absence requested 16#12: Applicator server is already active 16#13: Incorrect UNI-TE V2 transaction number	

Word number	High-order octet	Low-order octet	Data managed by
%MWk+1	16#01: Lack of resources for the processor 16#02: Lack of line resources 16#04: Line error 16#05: Length error 16#06: Faulty communication path 16#07: Addressing error 16#08: Application error 16#0B: Lack of system resources 16#0C: Inactive communication function 16#0D: Destination is absent 16#0F: Unconfigured infra-station or path routing problem 16#11: Uncontrolled address format 16#12: Lack of destination resources 16#14: Connection is not operating (example : Ethernet TCP/IP) 16#15: Lack of resources for the local path 16#16: Unauthorized access (example : Ethernet TCP/IP) 16#17: Network configuration is incoherent (example : Ethernet TCP/IP) 16#18: Connection is temporarily available 16#21: Applicator server is at a stop 16#30: Transmission error	16#FF: Message rejected	System
%MWk+2	Timeout		User
%MWk+3	Octet length : ● For a WRITE_VAR, initialize this word using the number of octets to be sent. ● For a EAD_VAR, when the request is terminated, this word contains the number of characters received in the table of words containing the received data.		

Note: the function may detect an error in the parameters before activating the exchange. In this case, the activity bit remains at 0, the record is initialized using the values corresponding to the default.

Examples of control parameters :

If the operation record is at 16#00 and the communication record is at 16#01 (cease exchange at timeout), this may mean that the port address has been incorrectly configured (1 instead of 0). In this case, it is a function block problem. Therefore, it is impossible to handle the CANopen services.

Accessing the CAN link layer PDUs

At a Glance

The communication function SEND_REQ provides access to the PDU s on the CAN link layer.

Syntax

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

```
SEND_REQ(ADR#y.1.SYS, 16#9F, %MWi:L, %MWj:L, %MWk:4)
```

The following table describes the various parameters of the function.

Parameter	Description
ADR#y.1.SYS	The address of the exchange destination device. • y: the processor's slot in the rack (0 or 1) • 1 : channel (always 1) • SYS: UNI-TE server of the PCMCIA card
16#9F	Function code
%MWi:L	Table of at least 4 words. • %MWi: action mode (101: send with local confirmation of the card) • %MWi+1 and %MWi+2: identifier at the level of the CANopen link layer; these two words form a double word which must contain a value between 0 and 536 870 911 (29 bits) **Note**: identifiers between 0 and 2047 (11 bits) are only accessed in write • %MWi+3 to %MWi+6 : PDU data, 8 maximum
%MWj:L	Not used (length set to 1)
%MWk:4	Exchange management parameters: four words identifying the address of the PL7 data used to control the function

Management parameters

The following table gives details of the %MWk:4 words.

Word number	Most significant byte	Least significant byte	Data managed by
%MWk	Exchange number	Activity bit	the system
%MWk+1	Operation report • correct response: 16#61 • incorrect response: 16#FD	Communication report	
%MWk+2	Timeout		user
%MWk+3	Length: initialization with the length of the table `%MWi:L` mandatory before sending the function. When the request is completed, this word contains the number of bytes received.		

IDENTIFICATION Request

At a Glance

This request is used to identify the CANopen bus master (TSX CPP 1•• PCMCIA card).

This request is made using the communication function SEND_REQ.

Syntax

The communication function syntax is as follows:

```
SEND_REQ(ADR#y.1.SYS, 16#0F, %MWi:L, %MWj:L, %MWk:4)
```

The following table describes the function's different parameters.

Parameter	Description
ADR#y.1.SYS	Address of the exchange destination entity. • y: processor slot number, 0 or 1 • 1: channel (always 1) • SYS: access to UNI-TE server from the PCMCIA card
16#0F	Request code
%MWi:L	Not used for the IDENTIFICATION function (length is 1)
%MWj:L	Table of words containing the card identification information. Length L must be 12.
%MWk:4	Exchange management parameters: four words identifying the address of the PL7 data used to control the IDENTIFICATION function

%MWj:L

The following table gives details of %MWj:L words.

Word number	Most significant byte	Least significant byte
%MWj	Product code: ● 16#04: Micro ● 16#05: Premium	16#FF
%MWj+1	Length of identification string: 16#0C	Number of BCD coded version (Version 1.0 coded 16#10)
%MWj+2	'S'	'T'
%MWj+3	' '	'X'
%MWj+4	'P'	'C'
%MWj+5	' '	'P'
%MWj+6	'0'	'1'
%MWj+7	16#00	'0'
%MWj+8	Indicator lamp status. The COM LED is coded on the first two bits and the ERR LED on the following two according to the following sequences: ● 0, 0: Off, ● 0, 1: Flashing, ● 1, 0: Permanently on	PCMCIA card state: ● 0: Absent, ● 1: Self-test, ● 2: Failure, ● 3: Ready, ● 4: Waiting, ● 5: Non- configured
%MWj+9	Product type: 16#02	Functional type: 16#2E
%MWj+10	Type of fault ● x0: Card in Test/Debug mode, ● x1: Bus fault or inactive bus, ● x2: Absent connection unit, ● x3: Self-testing or inaccessible card, ● x4: Reserved, ● x5: Card different to the one configured, ● x6: Card missing, ● x7: Error on at least one slave,	Catalog reference: 16#01
%MWj+11	-	16#00

Management Parameters

The following table gives details of the %MWk:4 words.

Word number	Most significant byte	Least significant byte	Data managed by
%MWk	Exchange number	Activity bit	the system
%MWk+1	Operation report, positive report 16#3F	Communication report	
%MWk+2	Timeout		you
%MWk+3	Length: initialization to 0 is mandatory before sending the function		

Diagnostic Command

At a Glance

Diagnostic commands are sent by the SEND_REQ function block:

In this case the SEND_REQ function is used for:

- obtain slave bus diagnostics,
- obtain the PCMCIA card version CANopen
- obtain state variables,
- obtain the history of bus errors.

Syntax

The syntax of the communication function is the following:

```
SEND_REQ(ADR#y.1.SYS,16#0031, %MWi:3, %MWj:L, %Mwk:4)
```

The following table describes the different function parameters.

Parameter	Description
ADR#y.1.SYS	Address of the exchange destination entity. ● y: processor slot in the rack (0 or 1) ● 1 : channel (always 1) ● SYS: UNI-TE server of the PCMCIA card
16#0031	Request Code

%MWi:3	Parameters of the request:
	● %MWi: diagnostic object type:
	  ● 1 ... 127 : Slave diagnostic 1 to 127
	  ● 128 : CANopen card version
	  ● 129 : Status of the CANopen card, the response is equivalent to the contents of the %IWy.1.i (see *Input Word Objects, p. 81*) status words.
	  ● 130 : History of messaging errors
	● %MWi+1: Starting address in the diagnostic table (Value by default 0). To enable partial access to the diagnostic table, specify a starting word in the table (Starting offset)
	● %MWi+2: Length in bytes of the diagnostics to be read, this length is generally two times the length of the responses table

%MWj:L	Reception table contains diagnostics data (see *Diagnostics data, p. 72*).
%MWk:4	Exchange management parameters: four word identifying the address of the PL7 data used to control the SEND_REQfunction.

Management parameters

The following table gives details of the %MWk:4 words.

Word number	Most significant byte	Least significant byte	Data managed by
%MWk	Exchange number	Activity bit	the system
%MWk+1	Operation report • positive response: 16#61 • incorrect response: 16#FD	Communication report	
%MWk+2	Timeout		User
%MWk+3	Length: number of response bytes **Note** Initialization compulsory in order to activate the function if the number of bytes emitted are 6.		

3.4 Debugging

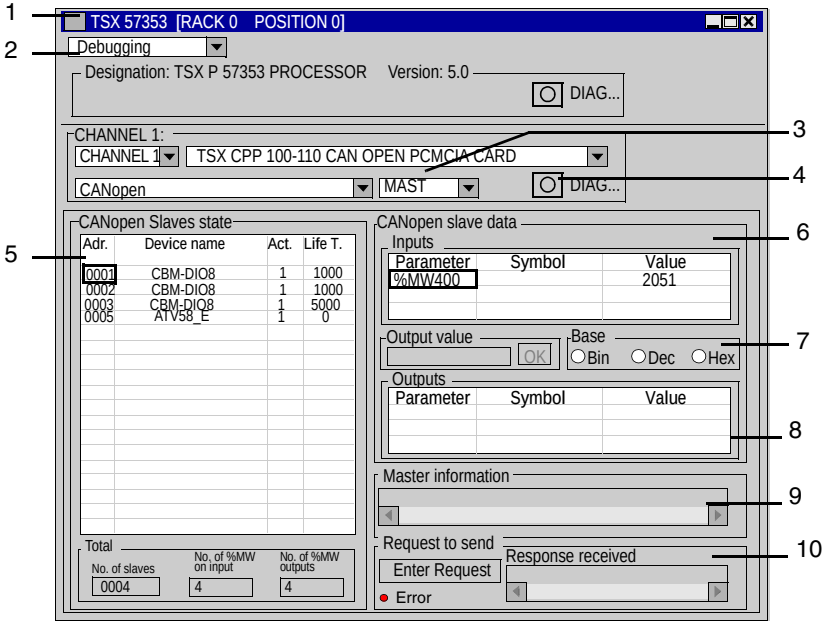
Description of the Debug Screen

At a Glance

The Debug function (selected in the configuration screen or by double-clicking on the PCMCIA TSX CPP 100-110 card for the PL7 configuration) is only available in online mode.

Illustration

The figure below illustrates the debug screen.



Description

The table below shows the different zones of the debug screen:

Address	Element	Function
1	Title bar	Indicates the product reference and the position of the module
2	Function selection	• Debug (available in online mode only) • Configuration
3	Channel zone	Indicates the characteristics of channel 1.
4	**DIAG** button	When a module fault is detected, this button is used to access information about the status of the module. **Note**: If the **DIAG** button is enabled (%Iy.MOD.ERR set to 1) and the exchanges with the slaves are normal, this signifies a latch of a bus fault at a given moment. To acknowledge this default, set the bit %QWy.1.0:x3=1 to 1 using the application or the animation table.
5	CANopen slave configuration	This zone displays all the CANopen bus slaves. A faulty slave is displayed in red; when this fault disappears, it is displayed in blue. Otherwise, it is displayed in black. Selecting a slave updates zones 6, 8, 9 and 10. **Act.** : indicates whether the slave was activated in the Sycon configuration (1=activated, 0=deactivated) **Life T.** : period of the Life Time or Heartbeat. **Note**: If the configuration file (see *How to Select a Configuration File, p. 50*) has not been found, the slaves will not be displayed.
6	Inputs	When a slave is selected, this zone contains the list of words which are associated with it on input.
7	Output value	When an output word in zone 8 is selected, its value may be modified by entering a new value then clicking on the **OK** button.
8	Outputs	When a slave is selected, this zone contains the list of words which are associated with it on output.
9	Information on ...	When a slave is selected (click in zone 5), this zone contains its last diagnostics message. To obtain information on the TSX CPP 100 card, simply click on the table header.
10	Request to send	When a slave is selected, this zone is used to send an SDO request. The parameters syntax is identical to the syntax used to perform SDO transfers using READ_VAR and WRITE_VAR (see *Syntax of the WRITE_VAR function, p. 56*) requests. Pressing on the **Enter Request** button makes the request input zones appear. The `Error` LED indicates a messaging error.

3.5 Diagnostics

At a Glance

Subject of this Section

This section describes the equipment and software diagnostics tools available for the TSX CPP 1•• PCMCIA card.

What's in this Section?

This section contains the following topics:

Topic	Page
Diagnostics Using the Status LEDs on the TSX CPP 100/110 PCMCIA Card	71
Diagnostics data	72
How to perform a diagnostic	75

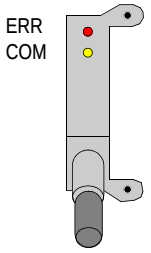
Diagnostics Using the Status LEDs on the TSX CPP 100/110 PCMCIA Card

At a Glance

LEDs on the card allow you to see the status of the card and the CANopen bus. In normal operation, the ERR LED is off and the COM LED is permanently on.

Illustration

The following figure indicates the position of the two LEDs, ERR and COM.



Diagnostics

Depending on the status of the LEDs, the diagnostics are as follows:

Status display LEDs		**Meaning**
ERR (red)	**COM** (yellow)	
Off	Off	Card has no power supply or configuration transfer in progress
	Blinking (Irregular)	No configuration on the card
	Blinking (Regular)	Card configured and ready, bus not active or no CANopen firmware
	On	Bus configured and active, no error
Permanently on	Off	Error detected, bus controller stopped
	Blinking	Card configured and ready, but unable to communicate with a remote peripheral device (e.g. CAN bus cable disconnected), or all the configured peripheral devices report an error. Error on card, configuration error, or synchronization error between the card and the PLC (for more information, consult the module status diagnostics data in the TSXDMCPP100 manual).
	On	Bus configured and active, at least one bus subscriber cannot be reached or signals an error

Diagnostics data

At a Glance

During diagnostics, the first data to be used is implicit exchange %IW words (see *Implicit Exchange Language Objects, p. 80*).

Elsewhere there is diagnostics data which can be consulted by writing some program lines onto the PLC.

The communication function SEND_REQ enables diagnostics to be carried out (see *Diagnostic Command, p. 66*):

- on the slave of its choice (1 to 127, one request per slave),
- on the PCMCIA card version (128),
- on the card state (129),

and is used to obtain the history report on the messaging error messages (130).

Note: code 129 enables the same information to be received as is contained in the implicit exchange input words (see *Input Word Objects, p. 81*).

The information provided comes from the PCMCIA card and is periodically updated.

The request reception table contains the information described in the following paragraphs.

Note: information is given in byte tables. Whilst taking into account the possibility of requesting all or part of this table, it is necessary to pay attention to the most significant and least significant words of the %MW:L table

Note: diagnostic information respects the CAN norm, you can find reference information on the site: `http://www.can-cia.de`.

Slave diagnostics

The following table describes the information received following a request for diagnostics on a slave (code 1 to 127).

Byte rank	Description
0	Device status bits: ● x0: No response ● x1: Overflow of error messages history table ● x2: Parameters error ● x3: Monitoring active device ● x4 to x6: Reserved ● x7: Deactivated
1 and 2	Complementary information read at bus start-up on the standard object 16#1000 (CAN standardization)
3 and 4	Number of profile read at bus start-up on the standard object 16#1000 (CAN standardization)
5	Slave status: ● 1 : Disconnected ● 2 : Connection in progress ● 3 : In preparation ● 4 : Ready ● 5 : Operating ● 127 : Being prepared or missing
6	Error code (see *Slave diagnostics codes, p. 92*) (code of the last error generated by the slave)
7	Number of blocks of urgent information on the slave (0 to 5). **Note**: these blocks are added to the end of this table. Details of a type of block are given in the following paragraph.

Description of an information block on the slave

The following table describes a type of information block.

Byte rank	Description
0 and 1	Error code
2	Value of error register, object 16#1001 of the slave (CAN standardization)
3 to 6	Register value of the specific manufacturer state, object 16#1002 (CAN normalization)
7	Reserved

Diagnostics on a card version

The following table describes the information received following a request for diagnostics on a PCMCIA card (code 128).

Byte rank	Description
0 to 7	Firmware version (character string)
8 to 10	Date of creation of the firmware version (BCD coded, DD.MM.YY)
11 to 13	Date of manufacture (BCD coded, DD.MM.YY)
14 to 17	Series number (BCD coded)
18 to 25	Protocol name (ASCII, without end of string character, e.g.: "CANopen")

Historical report of error messages

The following table describes the information received following a request for a historical report (code 130).

Byte rank	Description
0 and 1	Number of error blocks **Note**: these blocks are added to the end of this table. Details of a type of block are given in the following paragraph.
2 to 49	Contents of the error block (maximum 8 blocks)

Description of a history report information block

The following table describes a type of information block.

Byte rank	Description
0	Service code
1	ID concerned
2 and 3	Messaging error code (see *Error codes, p. 88*)
4 and 5	Detail on error code (see *History codes, p. 92*)

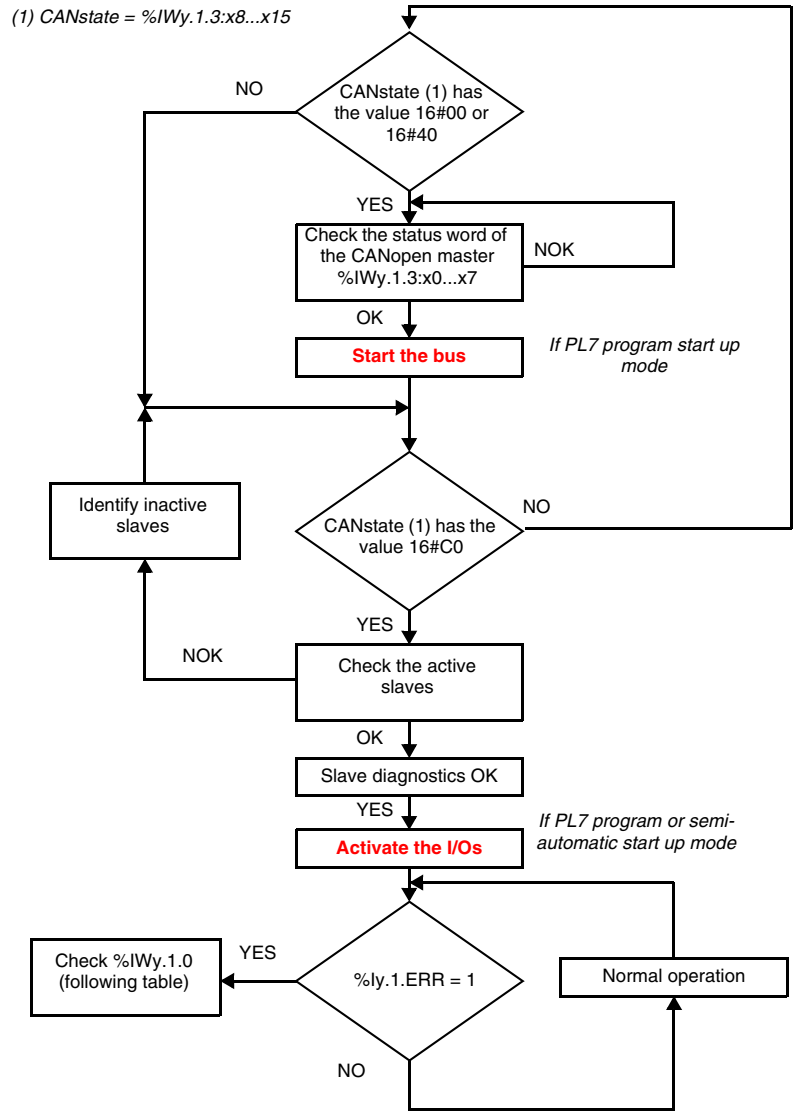
How to perform a diagnostic

At a Glance

You can start by using the PCMCIA card LEDs to search for faults on the CANopen bus. Next, you can use the procedure described below which details bus start up management and the checks to be carried out using the language objects (see *Language objects associated with the TSX CPP 100/110 card, p. 79*) provided by the PLC.

Procedure

The following table indicates the different phases of the procedure.



How to check %IWy.1.0

This table describes the actions to be carried out to obtain an accurate diagnostic using bits x8 to x15 of %IWy.1.0.

If	Then
x8 = 1	it is a configuration error. Check the detail of the error codes (see *Input Word Objects, p. 81*) in the words: ● %IWy.1.1 ● %IWy.1.2
x9 = 1	it is a PDO transfer error Contact Schneider technical support
x10 = 1	it is an SDO transfer error Check the detail of the error codes (see *Input Word Objects, p. 81*) in the words: ● %IWy.1.1 ● %IWy.1.2 Check the historical report of messaging errors (see *Historical report of error messages, p. 74*).

x11 = 1	it is a PCMCIA card error.
	● Check the error code details in %IWy.1.1.
	● Check the contents of %IWy.1.3:
	  ● x0: parameter error, the source of the error is indicated in %IWy.1.4
	  ● x1: the outputs are at zero after the failure of a slave (Autoclear ON), the source of the error is indicated in %IWy.1.4
	  ● x3: serious error, the card is not active on the bus
	  ● x7: faulty connection between the card and the TSX CPP ACC1 connection unit

x12 = 1	it is a bus error (bus not started or transmission error detected). ● Check the counter of %IWy.1.5 bus errors. If it is other than zero, check the line. ● Check the bus stops counter. If it is increasing, check the line and restart the bus. **Note**: in non-automatic start up mode, the bus start up bit is %QWy.1:x0. On the TSX CPP 110, this fault can also appear following disconnection of the bus at the TAP, or following the simultaneous switching on or off of all the bus slaves. The fault can be acknowledged by switching bit %QWy.1:X2 from 0 to 1.
x13 = 1	it is an error on a slave: communication error or I/Os not activated. ● Determine the last source of error contained in %IWy.1.4 ● Determine all the active slaves on the bus by consulting the bus status words %IWy.1.16 to %IWy.1.23 ● Perform a diagnostic of the faulty slaves using a diagnostics request (see *Diagnostic Command, p. 66*). **Note**: in non-automatic start up mode, the I/O start up bit is %QWy.1:x1.

If	Then
x14 = 1	it is an output error: the outputs are positioned in fallback conditions. ● Check that the PLC is in RUN mode ● Check that the task associated with the module is active ● Test the %IWy.1.0:x12 bit (Bus error) and the %IWy.1.0:x13 bit (Slave error). **Note**: in non-automatic start up mode, test the bits %QWy.1:x0 and %QWy.1:x1.
x15 = 1	a new diagnostic is available for one or more slaves. ● Determine these slaves by using the status words %IWy.1.16 to %IWy.1.23 ● Carry out a diagnostic (see *Diagnostic Command, p. 66*) for the slave(s) concerned

3.6 Language objects associated with the TSX CPP 100/110 card

At a Glance

Subject of this Section

This section describes the implicit and explicit language objects associated with the TSX CPP 1•• card.

What's in this Section?

This section contains the following topics:

Topic	Page
Implicit Exchange Language Objects	80
Language objects for explicit exchange	84
Explicit exchange management	85
Language objects associated with configuration	86
Error codes	88

Implicit Exchange Language Objects

At a Glance

This page describes all the implicit exchange (See: PL7 Micro/Junior/Pro; Communication applications; Volume 1) language objects for CANopen communication using the TSX CPP 1•• PCMCIA card, which can be displayed or modified by the application program.

Bit Objects

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

Object (1)	Function	Meaning
%Iy.MOD.ERR	Module error bit	This bit, when set to 1, indicates a module error (at least one of the channels has an error)
%Iy.1.ERR	Channel error bit	This bit, when set to 1, indicates a fault in the PCMCIA card.
Key		
(1)	Address y • y: corresponds to the processor's slot number in the rack (0 or 1 depending upon the power supply used)	

Input Word Objects

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

Object (1)	Function	Meaning
%IWy.1.0	Communication channel status word	Communication channel status bits: ● x0 = 1: Channel error, **OR** logic between the X1 and X14 bits that follow ● x1 to X7: Reserved ● x8 = 1: Incorrectly configured ● x9 = 1: Error in I/O exchange (PDO) ● x10 = 1: Messaging exchange error (SDO) ● x11 = 1: Card error (card missing or not ready) ● x12 = 1: Bus fault (at least one **bus error** type event has been generated) ● x13 = 1: Slave fault (a communication error has been detected with one or more slaves) ● x14 = 1: Output error (outputs are positioned at fallback conditions) ● x15 = 1: New slave diagnostics available (a new diagnostic is available for one or more slaves)
%IWy.1.1	Error word	This word contains a module error code (see *Module error codes, p. 88*) (last configuration or I/O error)
%IWy.1.2	Error word	This word contains a detailed module error code (see *Code 806 details, p. 91*) (last configuration or I/O error)

Object (1)	Function	Meaning

%IWy.1.3	CANopen master status word	CANopen communication master status bits:
		● x0 = 1: Parameters error
		● x1 = 1: Indicates that the outputs are at zero following the failure of a slave, Autoclear ON.
		● x2 = 1: No exchanges on the bus (no slave is communicating)
		● x3 = 1: Serious error, the card is not active on the bus
		● x4 = 1: One or more **bus error** events have been detected
		● x5 = 1: The processor has not yet authorized access to the card.
		● x6 = 1: Timeout exceeded when sending CAN messages
		● x7 = 1: Faulty connection between the card and the TSX CPP ACC1 connection unit (TSX CPP 100 only).
		● x8 to x15: These bits make up a byte whose value indicates the operating mode:
		  ● 16#00: offline mode
		  ● 16#40: bus in STOP mode
		  ● 16#80: outputs are in security mode (set to zero)
		  ● 16#C0: bus in RUN mode

%IWy.1.4	Status word for bus devices	The least significant byte (bits x0 to x7) contains the address of the slave which has generated the last error. The most significant byte (bits x8 to x15) contains the last error code.
%IWy.1.5	Counter status word	Counter of the number of bus errors
%IWy.1.6	Counter status word	Counter of the number of bus stops
%IWy.1.7	Counter status word	Counter of the number of timeouts on CAN messages
%IWy.1.8 to %IWy.1.15	Bus status words	For devices active on the bus, each bit set to 1 corresponds to an active slave on the bus (8 words of 16 bits = 128 bits, i.e. the master and 127 slaves)
%IWy.1.16 to %IWy.1.23	Bus status words	Diagnostics available on the bus, each bit set to 1 corresponds to a device for which diagnostics are available (8 words of 16 bits = 128 bits, i.e. the master and 127 slaves)

Object (1)	Function	Meaning
Key		
	Address y ● y: corresponds to the processor's slot number in the rack (0 or 1 depending upon the power supply used)	

Output Word Objects

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

Object (1)	Function	Meaning
%QWy.1.0	Operating mode control words	Control bits: ● x0 = 1: Activates the bus configuration x0 = 0: Deactivates the bus configuration This bit is only used when the bus startup is managed by the application ● x1 = 1: Activates data transfer on the bus x1 = 0: Deactivates data transfer on the bus This bit is used when the startup is semi-automatic or managed by the application ● x2 = 1: Initializes the error bits: ● I/O errors, ● messaging errors, ● history errors. ● x3 = 1: Initializes the PCMCIA card. This bit is used to warm start the card. This bit is used in the three start modes. ● x4 to x15: These bits are reserved. They are set to 0.
%QWy.1.1	Reserved	-
Key		
	Address y ● y: corresponds to the processor's slot number in the rack (0 or 1 depending upon the power supply used)	

Note: these implicit exchange output objects are also called command words. They are positioned by the application. Bits x2 and x3 of the %QWy.0 word are not automatically reset to zero.

Language objects for explicit exchange

At a Glance

This page describes the explicit exchange (See: PL7 Micro/Junior/Pro; Communication applications; Volume 1) language objects for CANopen communication using TSX CPP 1•• PCMCIA cards. Generally, these objects can be displayed or modified by the application program.

Note: objects %MWy.1.0 and %MWy.1.1 are used to manage explicit exchanges (see *Word objects, p. 85*).

Internal word

The following table describes the word %MWy.1.2, updated by the request READ_STS %CHy.1:

Object (1)	Function	Meaning

%MWy.1.2	Status of the PCMCIA card	• x0 = 1: Bus has a fault or is not initialized (in start-up mode managed by the application)
		• x1 = 1: Slave has an error, one or more slaves have errors or are not in RUN mode
		• x2 = 1: The TSX CPP ACC1 connection unit has an error or its cabling is faulty (TSX CPP 100 only).
		• x3 = 1: The PCMCIA card is:
		  • missing from its slot, or
		  • not ready, or
		  • has a serious error
		• x4 = 1: The PCMCIA card is:
		  • initializing, so not ready, or
		  • has an error, or
		  • cannot be accessed
		• x5 = 1: The card or protocol type is not recognized
		• x6 = 1: Error in I/O exchanges
		• x7 = 1: Configuration or parametering error
		• x8 to x15: Reserved, (value 0)

Key		
	Address y • y: corresponds to the processor's slot number in the rack (0 or 1 depending upon the power supply used)	

Explicit exchange management

At a Glance

This page describes all the language objects that manage explicit exchanges (See: PL7 Micro/Junior/Pro; Communication applications; Volume 1).

Word objects

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

Object (1)	Feature	Meaning
%MWy.1.0	Exchange in progress	• x0 = 1: Exchange in progress
%MWy.1.1	Exchange error	• x0 = 1: Error during exchange
Key		
	Address y • y: corresponds to the processor's slot number in the rack (0 or 1 depending upon the power supply used)	

Language objects associated with configuration

At a Glance

This page describes all the configuration language objects for CANopen communication with the TSX CPP 1•• PCMCIA card that can be displayed by the application program.

Internal Constants

The following table describes the internal constants:

Object	Function	Meaning
%KWy.1.0	Constant value used by the system	Least significant byte: 16#00 Most significant byte: 16#37
%KWy.1.1	Configuration bits	• Output fallback mode when PLC changes to STOP mode: x0 = 0: Reset x0 = 1: Maintain • x1 = 0: Load configuration via the terminal x1 = 1: Use Flash EEPROM configuration • Bus control on start-up: x2 = 0: Automatic x2 = 1: Via the application • I/O control on start-up: x3 = 0: Automatic x3 = 1: Via the application • Data exchange synchronization x4 = 0: MAST task x4 = 1: FAST task • x5 reserved • CANopen PCMCIA card watchdog x6 = 0: Activated x6 = 1: Deactivated • x7 to X15: Reserved
%KWy.1.2	Configuration bits	Size of the bus configuration in memory (in number of bytes)
%KWy.1.3	Configuration bits	Size of **input image** zone in memory (in number of words)
%KWy.1.4	Configuration bits	Size of **output image** zone in memory (in number of words)
%KWy.1.5	Configuration bits	Address of the start of the **input image** zone (%MW)
%KWy.1.6	Configuration bits	Address of the start of the **output image** zone (%MW)
Key		

Object	Function	Meaning
(1)	Address y ● y: corresponds to the position of the Processor in the rack (0 or 1 according to the selected power supply).	

Error codes

At a Glance

The following tables describe the different error codes that can arise in a CANopen configuration.

The explanations given in each table allow you to program the application in order to detect and correct any future operating anomalies more easily.

Module error codes

The following table describes the error codes located in words %IWy.1.1 (in the Description column) and %IWy.1.2 (in the Detail column).

Code	Description	Details
0	No error, operating correctly	
	Standard errors	
100	Invalid address (NULL pointer)	-
101	Invalid value	Value
102	Invalid object ID	ID
103	Invalid driver status	Status code
104	Size of read memory zone invalid	Memory size
105	Size of data to be written invalid	Size of data
106	Timeout	Timeout counter for an SDO transfer or loading status when downloading a configuration
107	Synchronization error	-
108	Stopped by the user	-
	Resource errors	
200	Memory overflow	-
201	Memory resources saturated	-
	Messaging errors	
300	Invalid source address	Value of address
301	Invalid destination address	Value of address
302	Invalid service	Service code
303	Invalid service class for a segment ID	Specified value
304	Primitive function of service invalid	Function code
305	ID of service called invalid	Specified value
306	Invalid communication gate	Gate number
307	Invalid bus device ID	Value of ID
308	Invalid SDO index	Value of index

Code	Description	Details
309	Invalid SDO sub-index	Value of sub-index
310	Remote error when executing a service	Error code
311	Invalid COB-ID	Value of COB-ID
312	Invalid type of transfer on the link layer	Code of the transfer requested: • 101 : send • 102 : receive • 103 : send and receive
	PCMCIA card errors	
600	Card missing	-
601	Card different to TSX CPP 100 / TSX CPP 110 detected	-
602	Card not ready to communicate	-
603	Card not yet in RUN mode	-
	PCMCIA card communication errors	
700	Error when sending a message to the card	-
701	Error when receiving a message coming from the card	-
702	Error when sending an output PDO to the card	-
703	Error when receiving an input PDO from the card	-
	Configuration errors	
800	Incorrect size of bus configuration data	Configuration data size
801	Size of input image data invalid	Size, in number of words, fixed at card start-up
802	Size of output image data invalid	Size, in number of words, fixed at card start-up
803	Overlap of memory areas reserved for inputs and outputs	Type of overlap: • 1 : the start of the input zone covers the end of the output zone • 2 : the start of the output zone covers the end of the input zone
804	Data loading zone not found	Type of zone: • 1 : global data • 2 : bus parameters • 3 : synchronization mode

Code	Description	Details
805	Invalid configuration data checksum (inconsistency of bus configuration data in Sycon mode)	Checksum of the bus configuration data located in the card's flash memory
806	Negative report on configuration downloading	Most significant byte: card error code (see *Code 806 details, p. 91*). Least significant byte: ● 16#00 : no loading in progress ● 16#01 : load request ● 16#02 : loading in progress ● 16#03 : loading completed ● 16#11 : transfer to the PC requested ● 16#12 : transfer to the PC in progress

Code 806 details

The following table describes the error codes, situated in the most significant byte of word %IWy1.2 and for value 806 of word %IWy1.1.

Code	Description
48	Timeout
52	Unknown zone code
53	Maximum memory size exceeded
55	Incorrect parameter
57	Sequence error in downloading
59	Downloaded data incomplete
60	Duplicate address
61	Size of PDO address table too big
62	Size of the bus device parameter zone too big
63	Mode of PDO transmission unknown
64	Size of PDO data too big
65	Transmission speed unknown
66	Synchro COB-ID outside limits
67	Synchro message Timer Preset value outside limits
68	Size of input data + offset greater than the maximum size of the input image zone
69	Size of output data + offset greater than the maximum size of the output image zone
70	Inconsistency between the PDO configuration and the PDO address table
71	Invalid PDO address table length
72	Invalid download data length
73	Urgent message COB-ID outside limits
74	Bus device monitoring message COB-ID outside limits
75	PDO length indicator outside limits
76	Size of SDO data too big

History codes

The following table describes the error codes, located in the fifth and sixth byte of the diagnostics history table.

Code	Description
3	Service rejected by the device
17	No response from the device
51	Length of the receive memory zone too long
53	Length of fragmented protocol data greater than the size of the buffer memory
54	Unknown function requested by the card driver
55	Bus device address outside limits
57	Sequence error during a fragmented transfer. The action is canceled.
200	card not configured

Slave diagnostics codes

The following table describes the error codes, located in the seventh byte of a slave's diagnostics table.

Code	Description
30	Error in slave monitoring
31	Change of status of a device on the bus, device unavailable
32	Sequence error in bus monitoring protocol
33	No response for a configured PDO
34	No response when configuring device
35	Profile of device configured different to profile of device present on the bus
36	Type of device configured different to type of device present on the bus
37	Unknown SDO response
38	Frame received longer than 8 bytes
39	Device not scanned or stopped (for example in Autoclear mode)

Sycon loading error codes

The following table describes the error codes that can occur while loading configuration or firmware onto the card using the XWAY driver.

Code	Description
0	No error, operating correctly
Standard errors	
8001	Driver inoperative
8002	Unknown event code from the driver
8003	Command code not recognized by the driver
8004	Command refused
8005	Another command is still active
8006	Command sent to an invalid device
Allocation errors	
8010	No device assigned
8011	Device already assigned
Communication errors	
8020	Sending of a service request when no device is connected
8021	Initialization of a connection when there is already one
8022	Time-out
8030	Driver status read error
8031	Error following the sending of a request on the network
8032	Outbox still busy
8033	Network response error
8034	No response available from the inbox
8035	Input/output data transfer error
Driver initialization errors	
8080	Parametering errors
8081	General driver initialization error
Multitask processing errors	
-1	Work task not created
-2	Invalid task pointer or synchronized object
-3	No synchronization event has been created

Examples of Configuration with SyCon Tool

4

At a Glance

Aim of this Chapter

This chapter describes, using examples, the installation software of the CANopen bus configuration with the help of SyCon tool (V2.8).

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
4.1	Standard configuration	96
4.2	Multi-master Configuration	112

4.1 Standard configuration

At a Glance

Aim of this Section

This section describes, using examples, the configuration software of a CANopen bus managed by one master.

What's in this Section?

This section contains the following topics:

Topic	Page
Example Description	97
Declaration of CANopen Master	98
CANopen Bus Configuration	100
Declaration of slave 7	101
Configuration of slave 7	103
Declaration of slave 8	107
Configuration of slave 8	108
Checking the configuration	109

Example Description

At a Glance

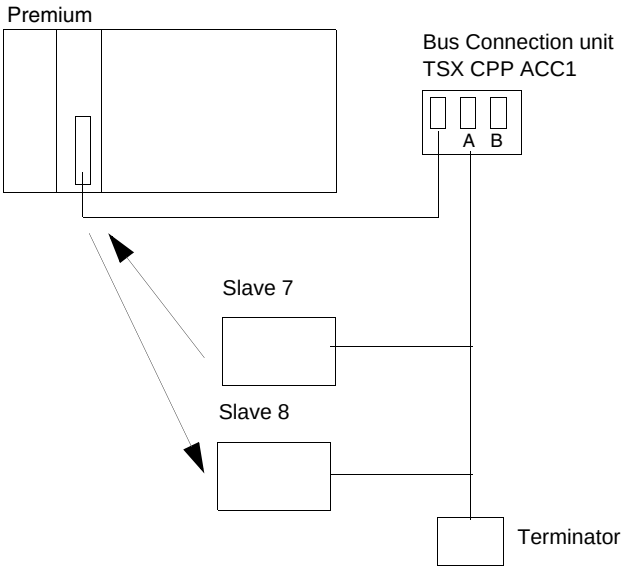
This example is given for educational purposes, It allows you to follow the different stages in the configuration of a CANopen structure made up of:

- a TSX CPP 100 master module installed in a premium PLC,
- for a TSX CPP ACC1 connection unit,
- with two slave devices:
  - a module with 8 inputs (slave7),
  - a module with 8 outputs (slave 8),

The speed of the bus transmission is fixed at 1Mbit/s.

Illustration

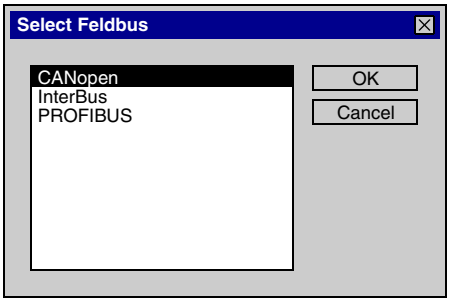
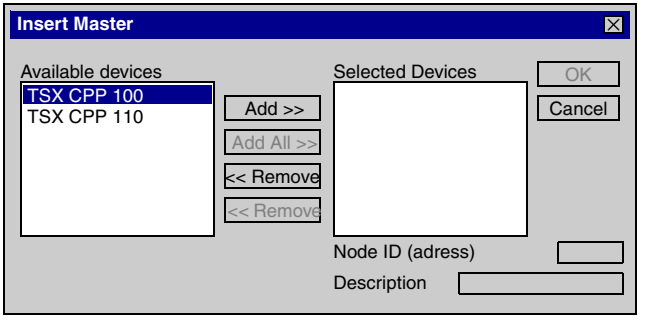
The illustration below is a diagram of the structure used in the example.

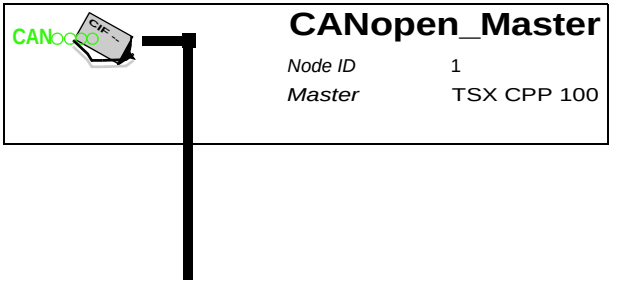


Declaration of CANopen Master

Procedure

The table below shows the different stages when declaring the master CANopen.

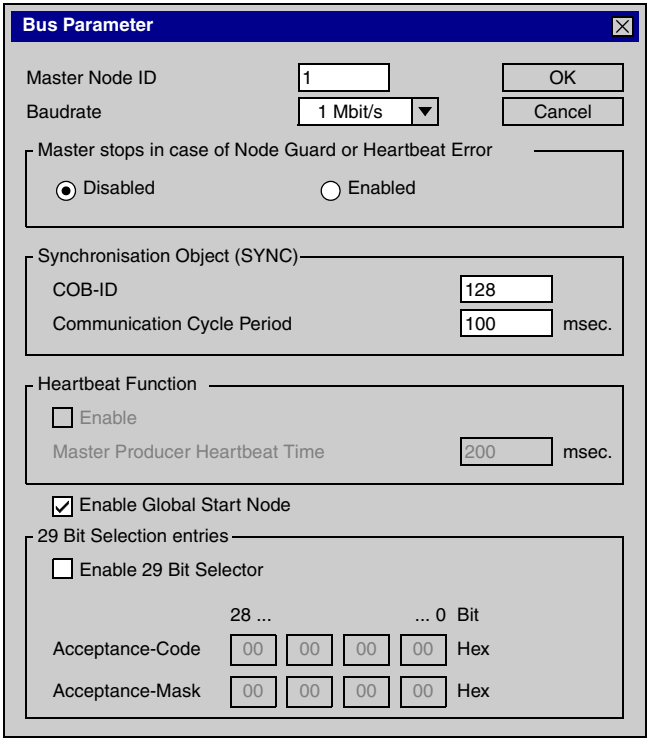
Step	Action
1	From the TSX CPP100 card configuration screen in Premium PL7, click on the icon  **Result:** The SyCon tool will appear on the screen.
2	Select the command **File** → **New** **Result:** The following screen appears: 
3	Select **CANopen** then confirm by clicking **Ok**. **Result:** A blank structure will appear on the screen.
4	Select the command **Insert** → **Master**. **Result:** The following screen appears: 

Step	Action
5	• Select **TSX CPP 100**, • click on **Add**, • enter a name, which represents the master device, in the **Description** field, **Note:** The name should contain neither spaces nor letters with accents and should consist of no more than 32 letters. • confirm by clicking **Ok**. **Result:** The following structure appears: 

CANopen Bus Configuration

Procedure

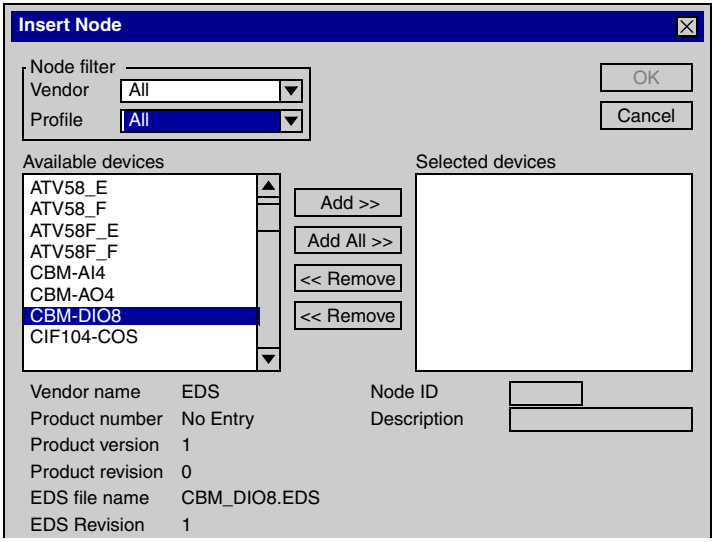
The table below shows the different stages for declaring the master CANopen.

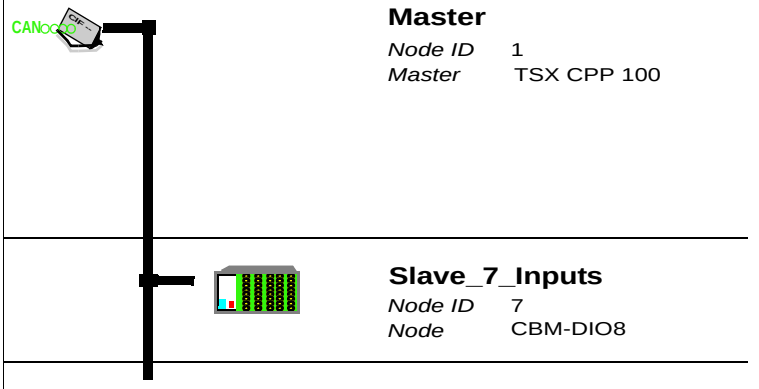
Step	Action
1	Select the command **Settings** → **Bus Parameters**. **Result:** The following screen appears: 
2	Configure: ● Baudrate at 1 Mbit/s, ● the value of **SYNC COB-ID** at 128 (default value), ● the Communication Cycle Period is at 100ms.
3	Select **Disabled** in **Master stops in case of Node Guard or Heartbeat Error**.
4	Select **Enable Global Start Node**.
5	Confirm by clicking **Ok**.

Declaration of slave 7

Procedure

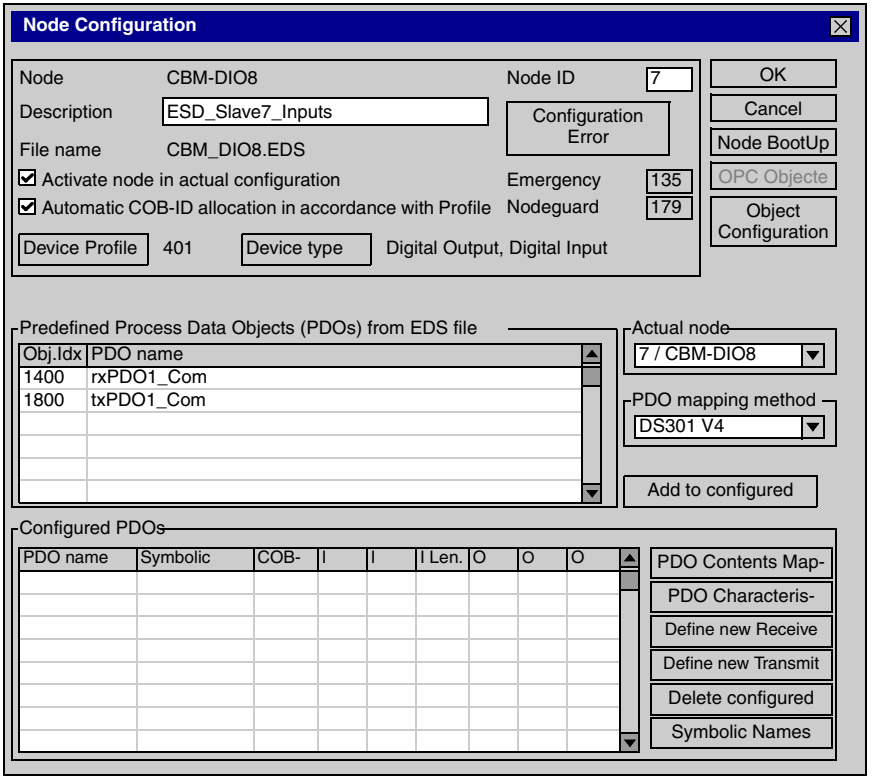
The table below shows the different stages to declare the slave 7.

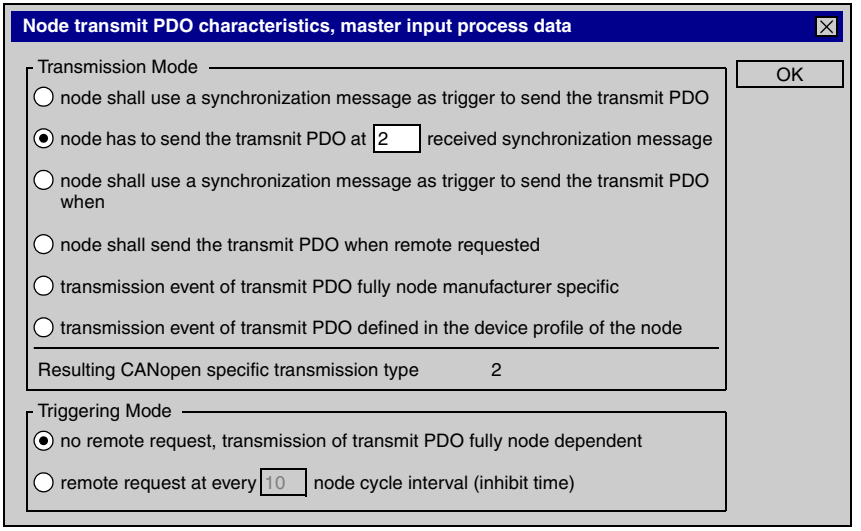
Step	Action
1	Select the command **Insert** → **Node**. **Result:** A cursor appears:
2	Place the cursor on the bus outside of the text frame that defines the master. **Result:** The following screen appears: 

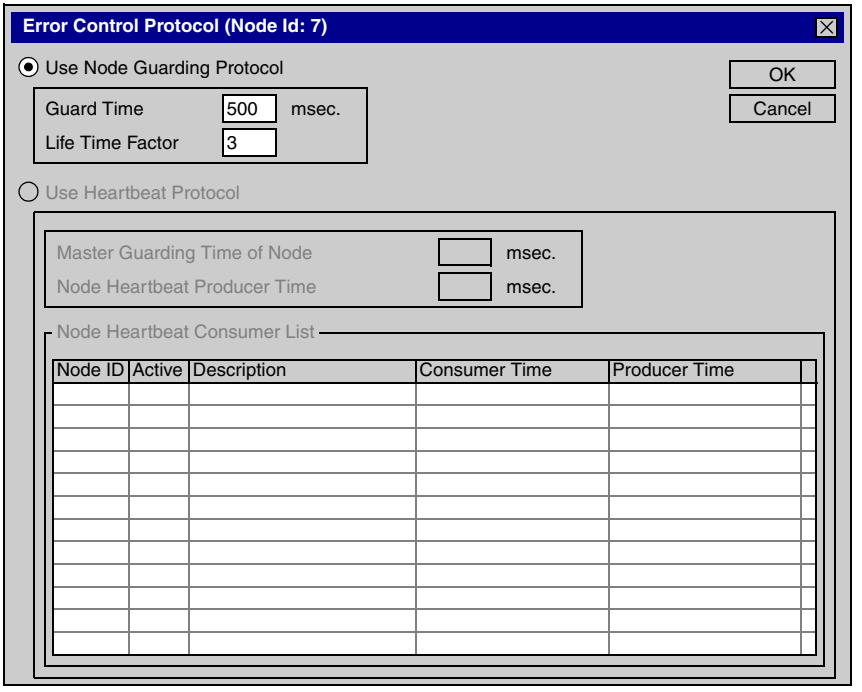
Step	Action
3	Select: ● the vendor name (ESD for example), ● the slave profile (401 for example), ● in the list of slaves, the slave type(e.g. CBM-DI08) then click on **Add**, ● 7 in the field **Node ID** then enter a name of a slave device in the **Description** field, ● confirm by clicking **Ok**. **Result:** The following structure appears: 

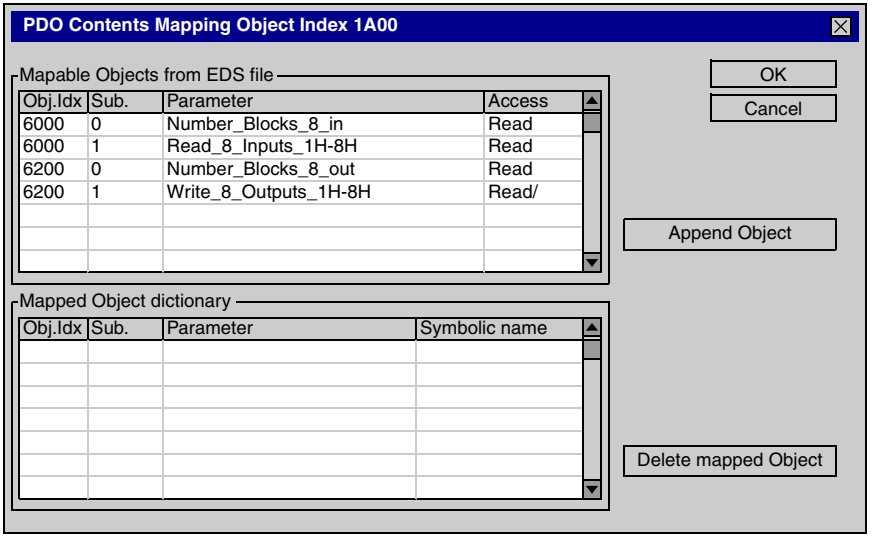
Configuration of slave 7

Procedure The table below shows the different steps for configuring slave 7.

Step	Action
1	Double click on slave 7. **Result:** The following screen appears: 

Step	Action
2	Select the object **1800 txPD01_Com** then click **Add to configured PDOs**. **Result:** The following window appears: 
3	Modify the transmission frequency as necessary, then confirm by clicking **Ok**.

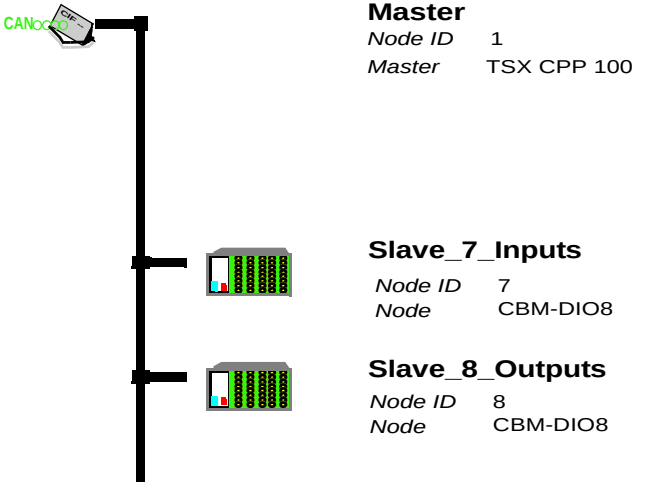
Step	Action
4	Click on button **Configuration Error Control Protocol**. **Result:** The following window appears:  Change **Life Time Factor** to **3** then confirm clicking **OK**.
Note	if one of the "Guard Time" or " Life Time Factor" parameters is set to 0 in the slave configuration screen, the outputs will not return to their fallback values in the event of a break in connection (*) between this slave and cards TSX CPP 100 / TSX CPP 110. (*) this disconnection can be caused by: ● a processor fault, or a break in its power supply, ● disconnection of the cable relaying the PCMCIA card and its TAP, ● disconnection of the slave from the bus, ● a faulty bus cable, ● a Sycon "Reset" command (Online → Firmware / Reset), ● a SyCon load configuration command (Online → Download), ● a command for firmware download to the card via SyCon (Online → Firmware Download).

Step	Action
5	Select **PDO_1800** in the **Configured PDOs** field, then click **PDO Content Mapping**. **Result:** The following window appears: 
6	Select **Read_8_Inputs_1H-8H (Idx.Obj. 6000, Sub.Idx. 1)**, click **Append Object** then confirm by clicking **Ok**.
7	Confirm configuration screen by clicking **Ok**.

Declaration of slave 8

Procedure

The table below shows the different stages to declare the slave 8.

Step	Action
1	Select the command **Insert** → **Node**.
2	Place the scale indicator on the bus outside of the text frame that defines the master.
3	Select: ● the vendor name (ESD for example), ● the slave profile (401 for example), ● in the list of slaves, the slave type (e.g. CBM-DI08) then click on **Add**, ● Enter 8 in the **Node ID** field then enter a name of a slave device in the **Description** field, ● confirm by clicking **Ok**. **Result:** The following structure appears: 

Configuration of slave 8

At a Glance

The configuration of slave 8 is carried out in the same way as the configuration of slave 7 (see *Configuration of slave 7, p. 103*).

The main difference is in selecting the outputs instead of the inputs.

Procedure

The table below shows the different steps for configuring slave 8.

Step	Action
1	Double click on slave 8. **Result:** The configuration screen appears
2	Select the object **1400 txPD01_Com** then click **Add to configured PDOs**. **Result:** The PDO characteristics window appears
3	Modify the transmission frequency as necessary, then confirm by clicking **Ok**.
4	Click on button **Configuration Error Control Protocol**. Change **Life Time Factor** to **3** then confirm clicking **OK**.
Note	if one of the "Guard Time" or " Life Time Factor" parameters is set to 0 in the slave configuration screen, the outputs will not return to their fallback values in the event of a break in connection (*) between this slave and cards TSX CPP 100 / TSX CPP 110. (*) this disconnection can be caused by: ● a processor fault, or a break in its power supply, ● disconnection of the cable relaying the PCMCIA card and its TAP, ● disconnection of the slave from the bus, ● a faulty bus cable, ● a Sycon "Reset" command (Online → Firmware / Reset), ● a SyCon load configuration command (Online → Download), ● a command for firmware download to the card via SyCon (Online → Firmware Download).
5	Select **PDO_1400** in the **Configured PDOs** field, then click **PDO Contents Mapping** .
6	● Select **Write_8_Ouputs_1H-8H (Idx.Obj. 6200, Sub.Idx. 1)**, ● click **Append Object**, then ● confirm by clicking **Ok**.
7	Confirm configuration screen by clicking **Ok**.

Checking the configuration

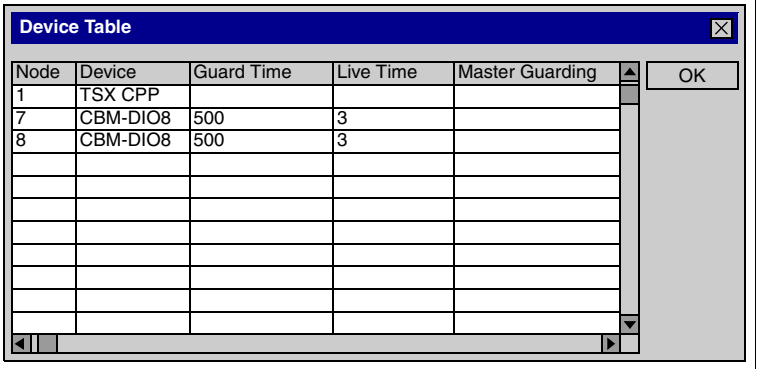
At a Glance

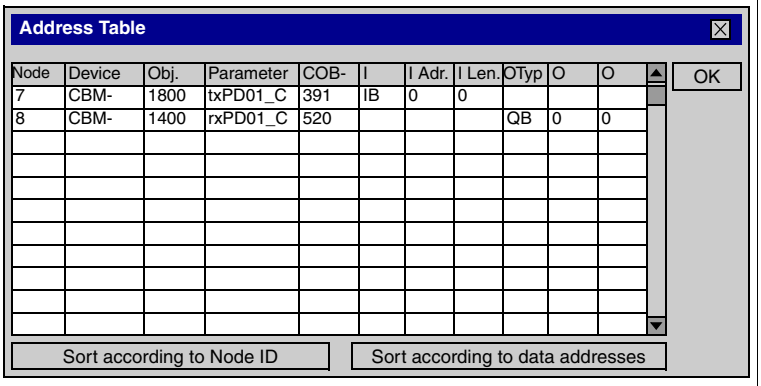
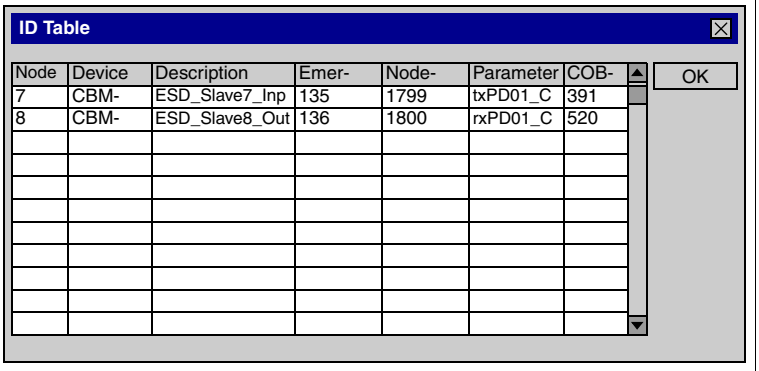
The SyCon tool allows you to monitor work carried out during this example. Access to the various display tables is shown below.

Note: after monitoring your bus configuration, do not forget to save it in a .CO file. This will be used during loading of the PL7 application or during direct loading in the TSX CPP 1•• card.

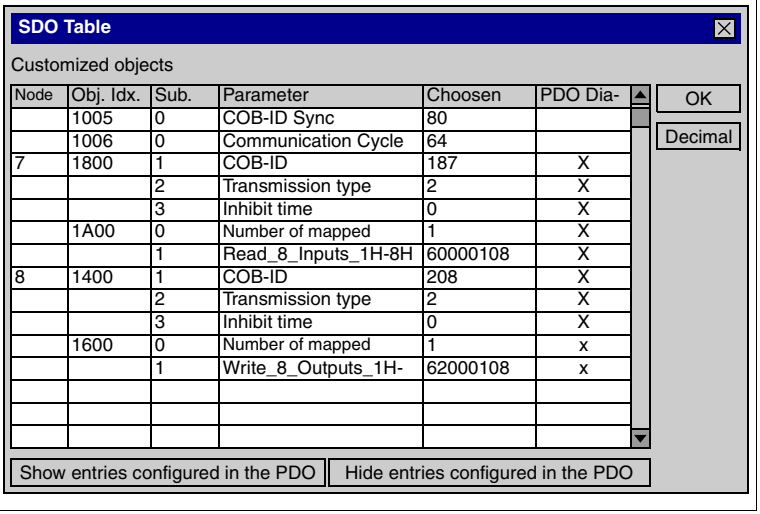
Procedure

The table below shows the main bus configuration verification steps using the display tables.

Step	Action
1	Select the command **View** → **Device Table** **Result:** the list of configured devices appears on the following screen. 

Step	Action
2	Select the command **View → Address Table** **Result:** a list of configured slaves and their associated specifications data appear on the following screen. 
3	Select the command **View → ID Table** **Result:** a list of configured devices and their identifiers appears on the following screen. 

Step	Action
4	Select the command **View → SDO Table** **Result:** a list of configured slaves and their SDO specifications appears on the following screen.



4.2 Multi-master Configuration

At a Glance

Aim of this Section

This Section describes, using examples, the configuration software of a CANopen bus, which has two coexisting masters.

This configuration is carried out with the help of the SyCon configuration tool.

What's in this Section?

This section contains the following topics:

Topic	Page
Example Description	113
Declaration of CANopen Master A and slaves 7 and 8	115
Declaration of Master B CANopen	116
Master B CANopen Bus Configuration	118
Declaration and Configuration of Slaves 9 and 10	119
Declaration and Configuration of Slave 7 for Master B	120
Declaration and Configuration of Slave 8 for Master B	125

Example Description

At a Glance

This example is given for educational purposes. It allows you to follow the different stages in the configuration of a CANopen structure made up of:

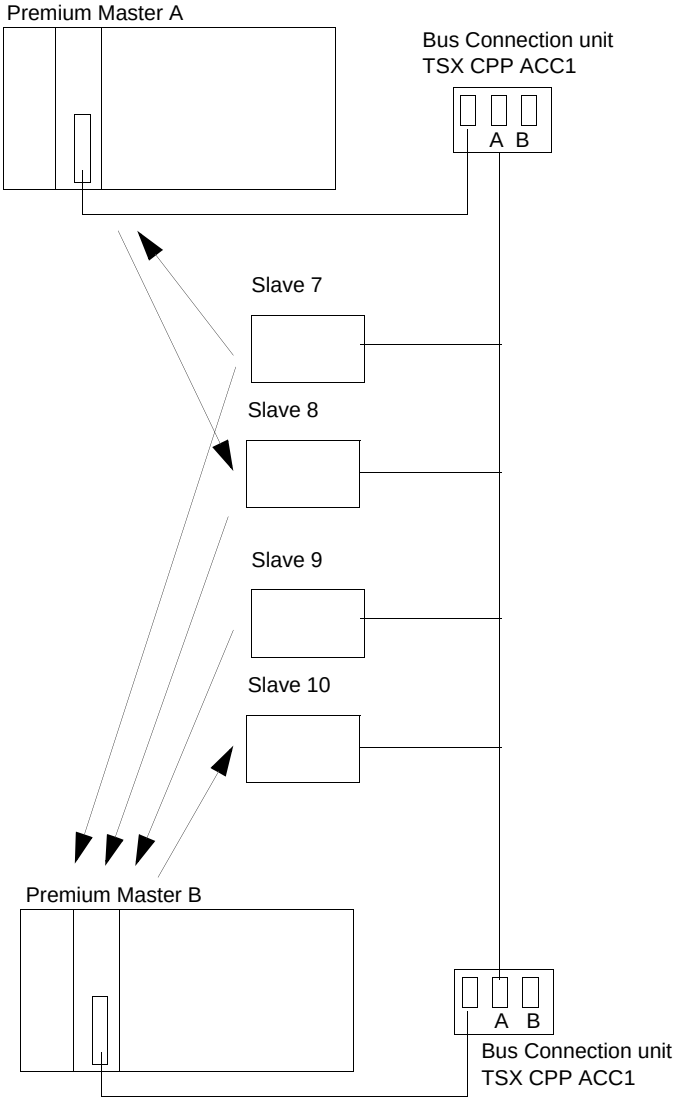
- a TSX CPP 100 master module(called master A) installed in a Premium PLC,
- a TSX CPP 100 master module (called master B) installed in a Premium PLC,
- on two TSX CPP ACC1 connection units,
- with four slave devices:
  - a module with 8 inputs (slave 7),
  - a module with 8 outputs (slave 8),
  - a module with 8 inputs (slave 9),
  - a module with 8 outputs (slave 10).

The speed of the bus transmission is fixed at 1Mbit/s.

Requested functions of masters A and B:

- Master A manages slaves 7 and 8.
- Master B manages slaves 9 and 10.
- Master B is listener on slaves 7 and 8.

Illustration The illustration below is a diagram of the structure used in the example.



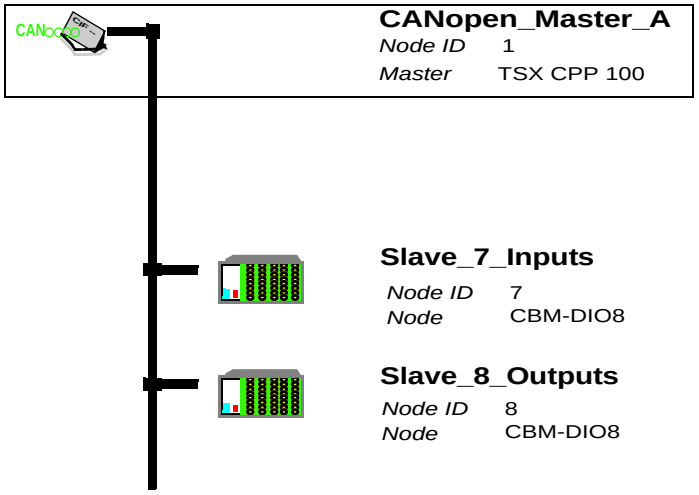
Declaration of CANopen Master A and slaves 7 and 8

At a Glance

The declaration of bus CANopen A configuration as well as the declaration and slave 7 and 8 configuration are carried out in the same way as the example of the standard bus (see *Standard configuration, p. 96*).

Result

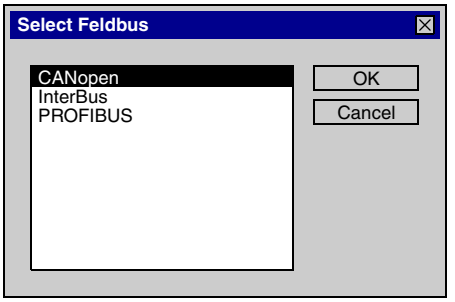
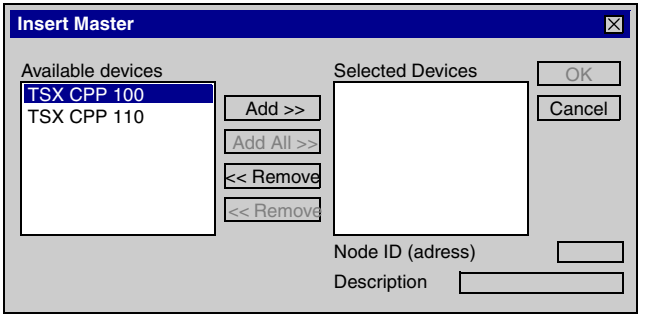
The figure below illustrates the CANopen Master A configuration.

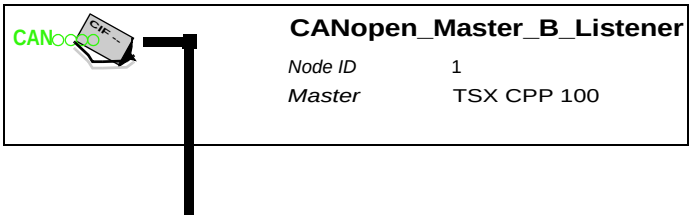


Declaration of Master B CANopen

Procedure

The table below shows the different stages to declare the master CANopen.

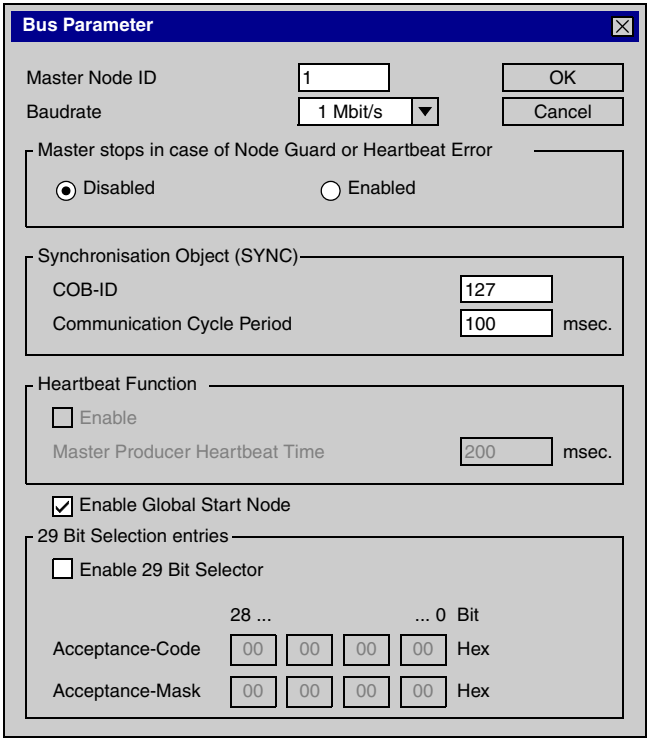
Step	Action
1	From the TSX CPP100 card configuration screen in PL7, click on the icon  **Result:** The SyCon tool will appear on the screen.
2	Select the command **File** → **New** **Result:** The following screen appears: 
3	Select **CANopen** then confirm by clicking **Ok**. **Result:** A blank structure will appear on the screen.
4	Select the command **Insert** → **Master**. **Result:** The following screen appears: 

Step	Action
5	• Select **TSX CPP 100**, • click on **Add**, • enter a name, which represents the master device, in the **Description** field, **Note:** The name should contain neither spaces nor letters with accents and should consist of no more than 32 letters. • confirm by clicking **Ok**. **Result:** The following structure appears: 

Master B CANopen Bus Configuration

Procedure

The table below shows the different stages for declaring the CANopen bus.

Step	Action
1	Select the command **Settings** → **Bus parameters**. **Result:** The following screen appears: 
2	Configure: • speed at 1 Mbit/s, • the value of **SYNC COB-ID** at 127 (every master on the same bus must have a different SYNC COB-ID), • the Communication Cycle Period is at 100ms.
3	Select **Disabled** in **Master stops in case of Node Guard or Heartbeat Error**.
4	Select **Enable Global Start Node**.
5	Confirm by clicking **Ok**.

Declaration and Configuration of Slaves 9 and 10

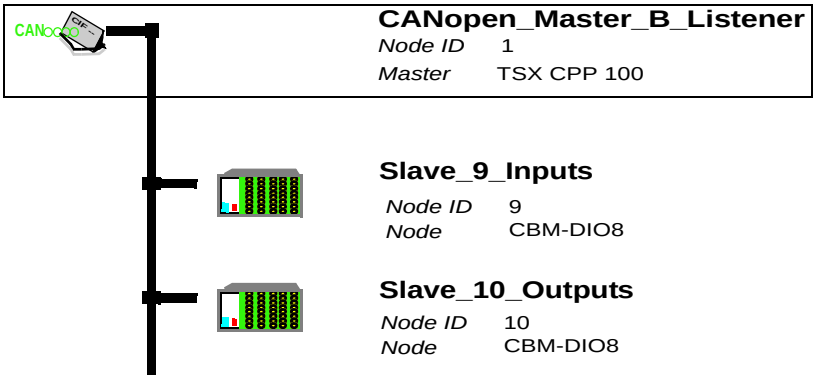
At a Glance

The declaration and configuration of slaves 9 and 10 is carried out in the same way as it is for slaves 7 and 8 for the standard bus, only the node numbers and the description of each slave changes:

- Declaration of slave 9 (see *Declaration of slave 7, p. 101*),
- Configuration of slave 9 (see *Configuration of slave 7, p. 103*),
- Declaration of slave 10 (see *Declaration of slave 8, p. 107*),
- Configuration of slave 10 (see *Configuration of slave 8, p. 108*),

Result

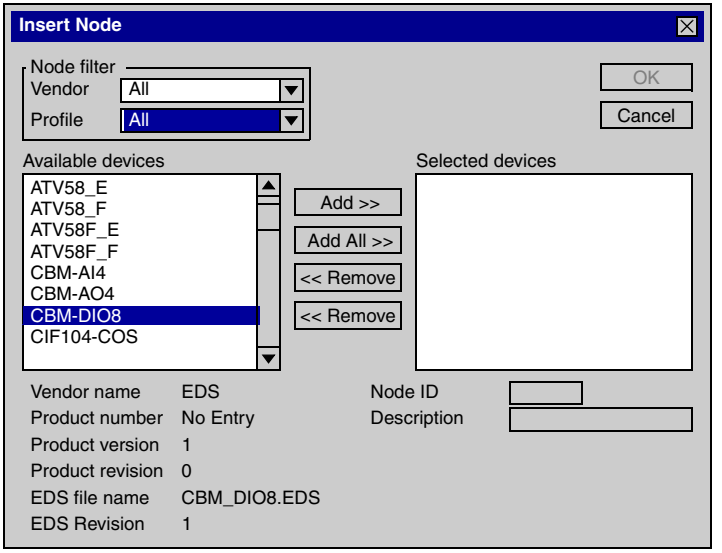
The structure below illustrates the configuration of slaves 9 and 10 on master B.



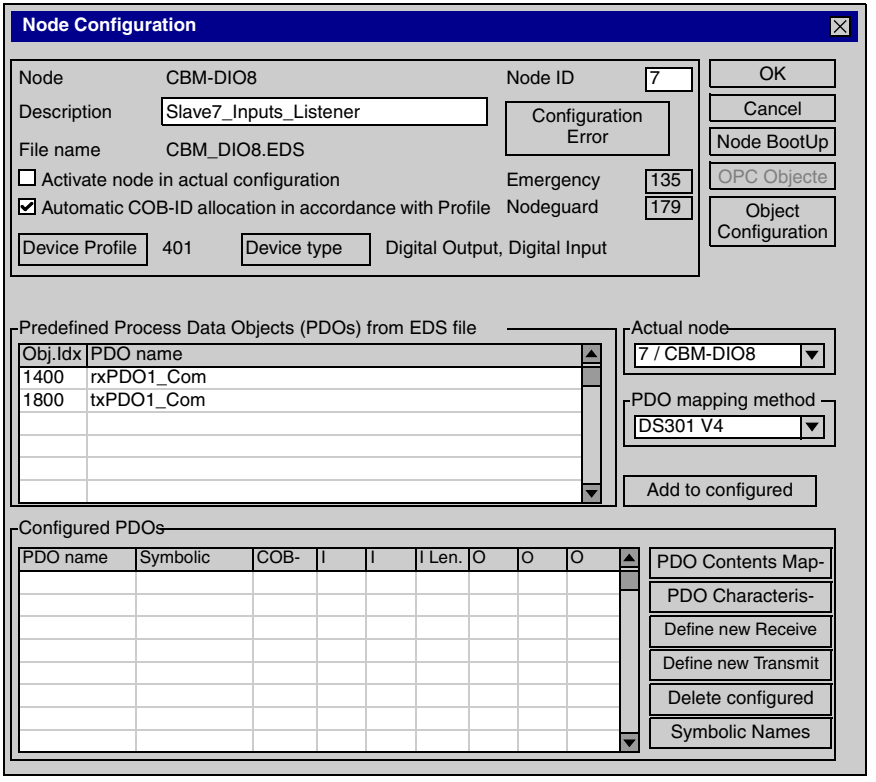
Declaration and Configuration of Slave 7 for Master B

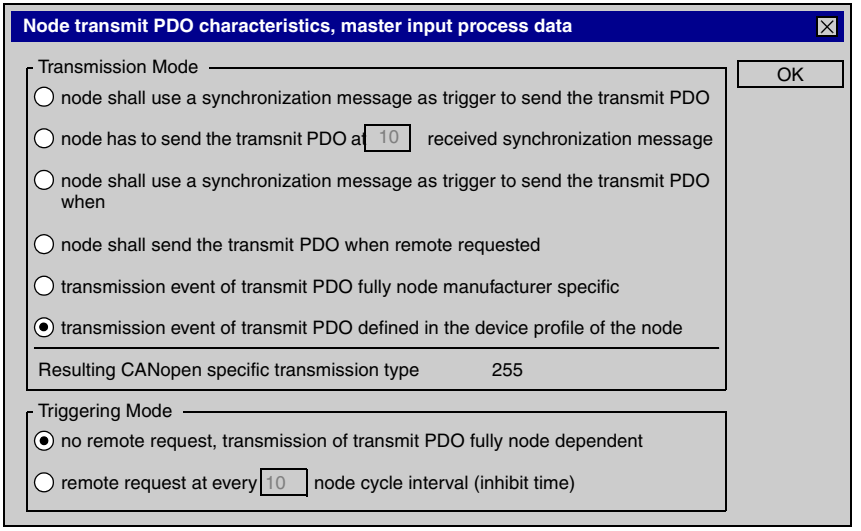
How to declare slave 7

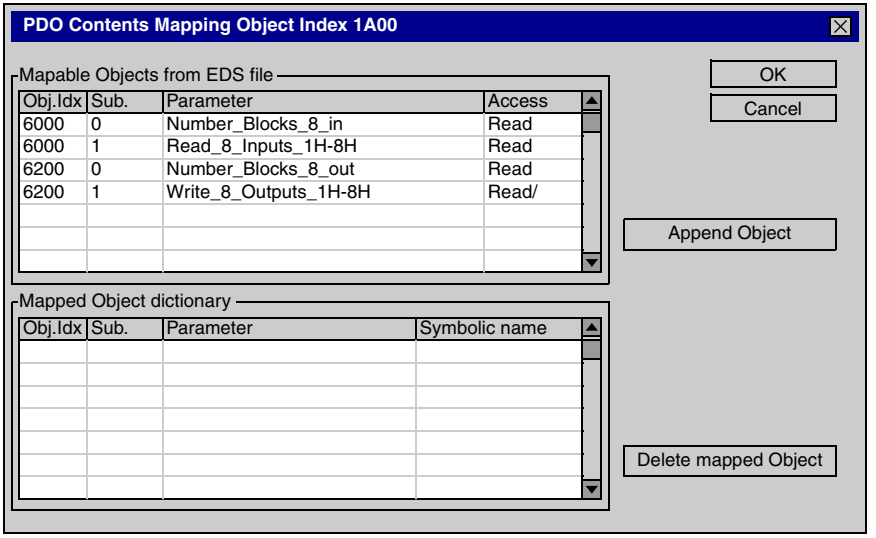
The table below shows the different stages to declare the slave 7.

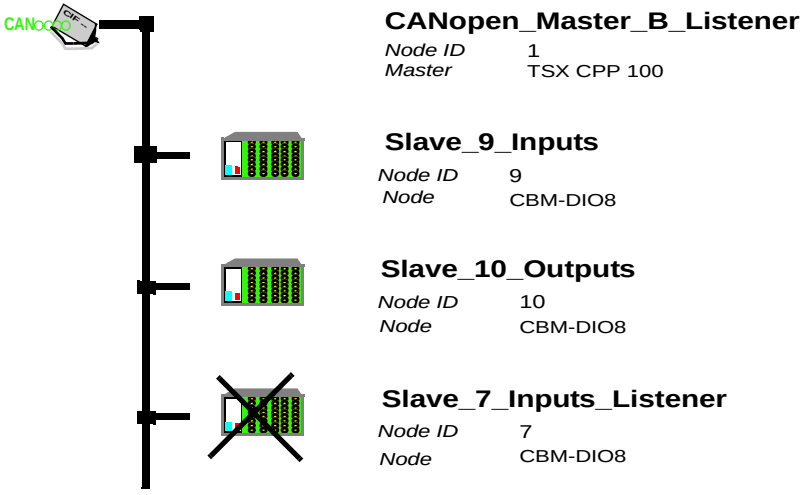
Step	Action
1	Select the command **Insert** → **Node**. **Result:** A cursor appears:
2	Place the cursor on the bus outside of the text frame that defines the master. **Result:** The following screen appears: 
3	Select: • the vendor name (ESD for example), • the slave profile (401 for example), • in the list of slaves, the slave type (e.g. CBM-DI08) then click on **Add**, • 7 in the field **Node ID** then enter a name of a slave device in the **Description** field, • confirm by clicking **Ok**.

How to configure slave 7 The table below shows the different stages to configure the slave 7.

Step	Action
1	Double click on slave 7. **Result:** The following screen appears: 
2	Uncheck the **Activate node in actual configuration** box.

Step	Action
3	Select the **1800 txPD01_Com** object then click on the **Add to configured PDOs** button. **Result:** The following window appears: 
4	• Select **transmission event of transmit PDO defined in the device profile of the node**, • Select **no remote request, transmission of transmit PDO fully node dependent**, • confirm by clicking **Ok**.

Step	Action
5	Select **PDO_1800** in the **Configured PDOs** field then click on the key **PDO Contents Mapping** . **Result:** The following window appears: 
6	Select **Read_8_Inputs_1H-8H (Obj.Idx.6000, Dub.Idx. 1)**, click on **Append Object** then confirm by clicking **Ok**.

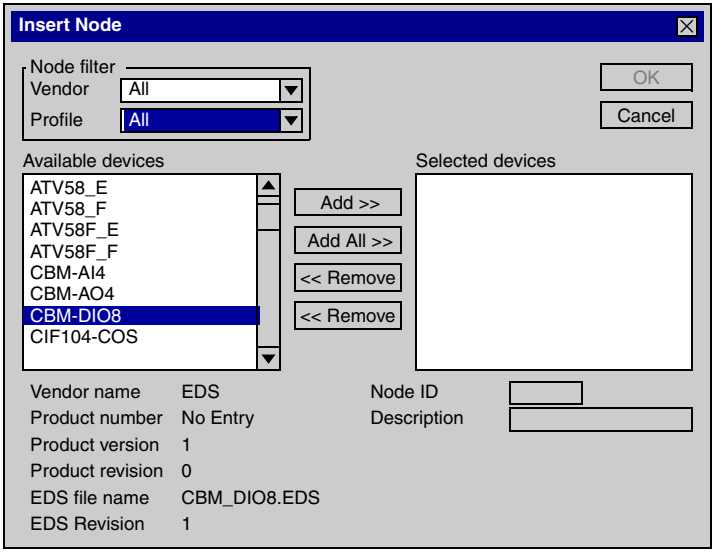
Step	Action
7	Confirm configuration screen by clicking on **Ok**. **Result**: The following structure appears: 

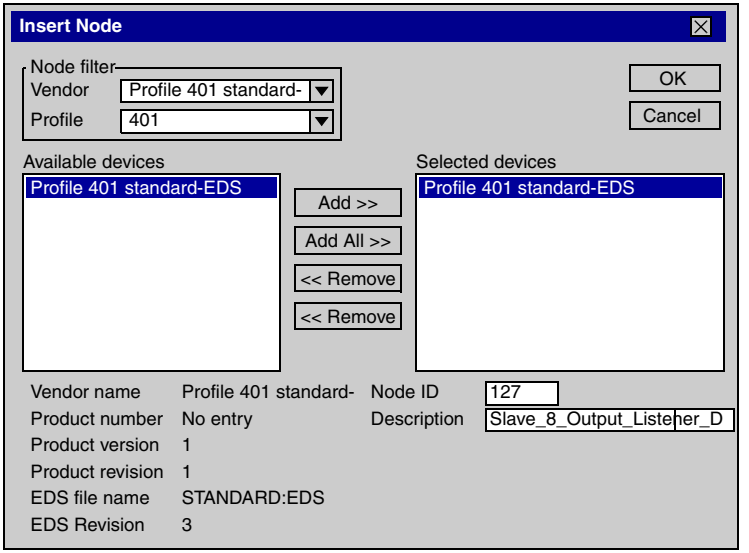
Declaration and Configuration of Slave 8 for Master B

How to declare slave 8

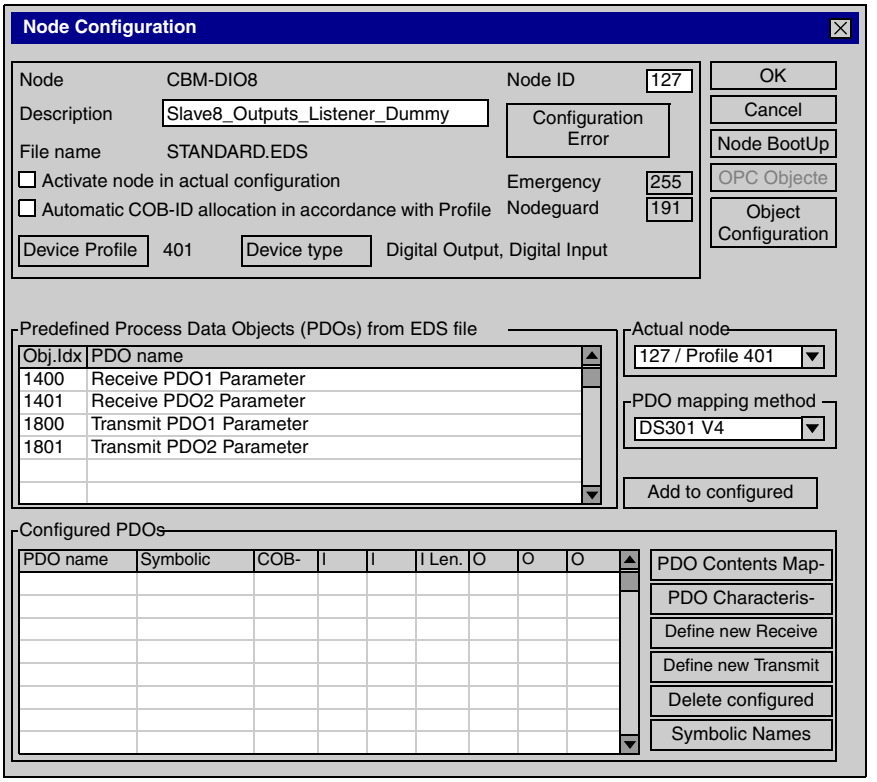
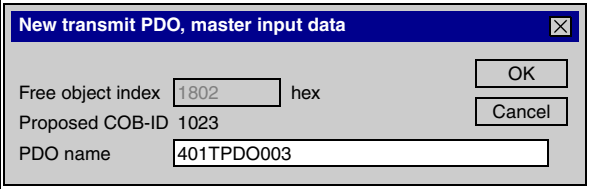
The table below shows the different stages to declare the slave 8.

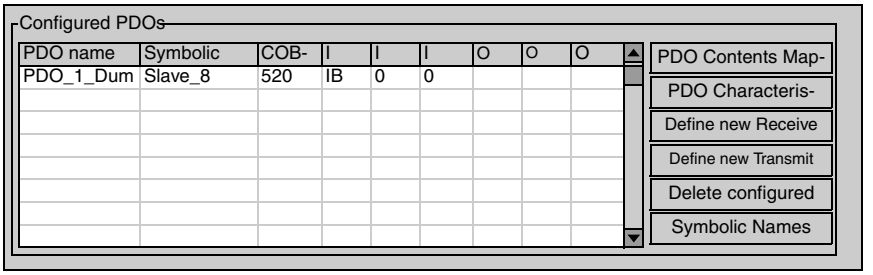
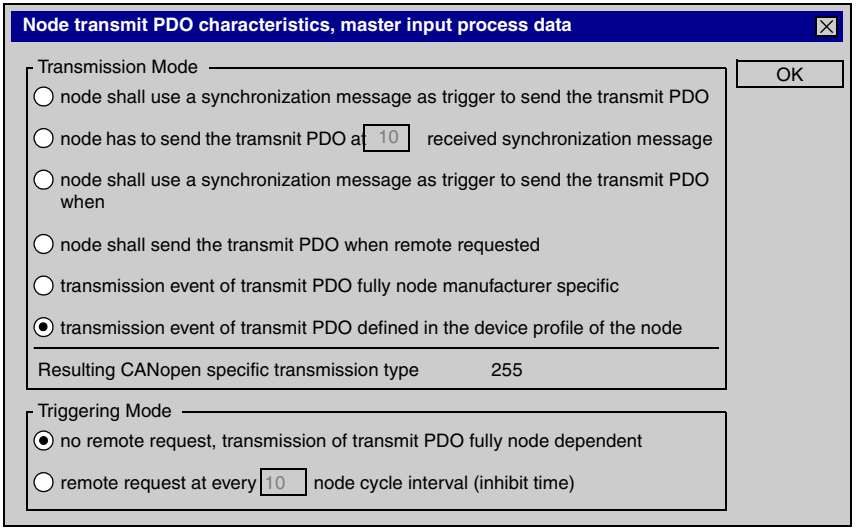
Step	Action
1	Select the command **Insert** → **Node**. **Result:** A cursor appears:
2	Place the cursor on the bus outside of the text frame that defines the master. **Result:** The following screen appears:

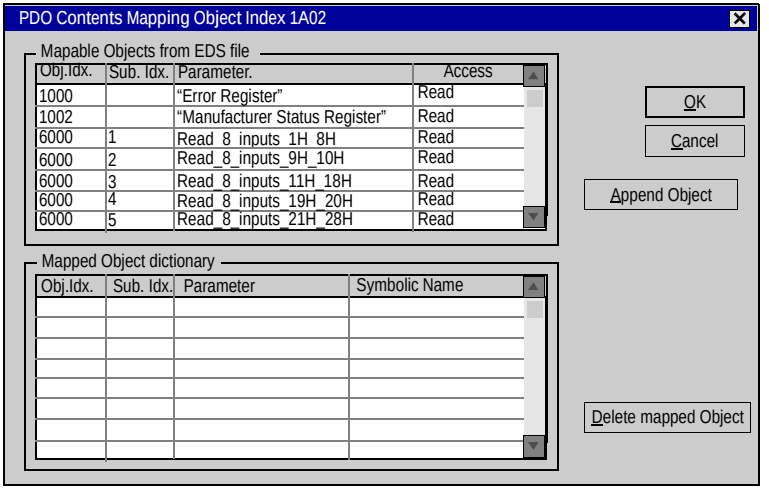


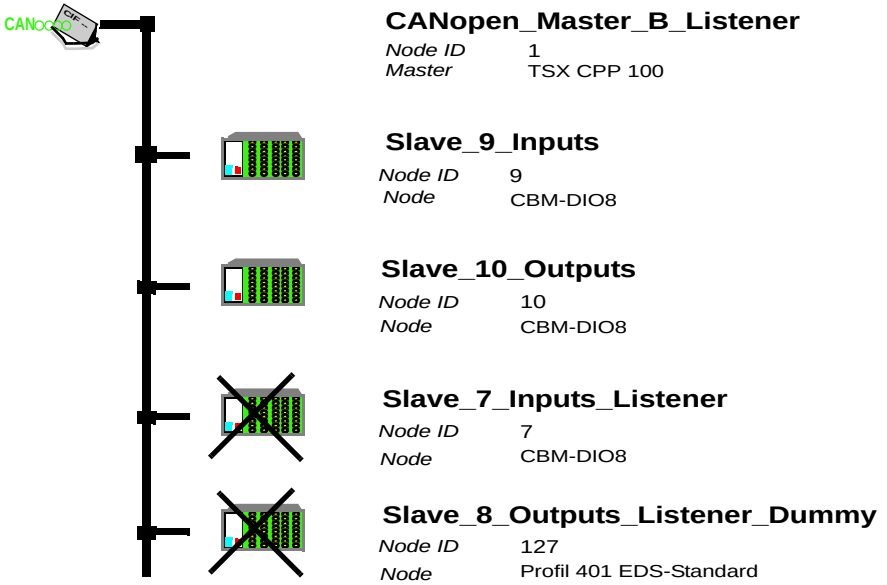
Step	Action
3	Select: ● **Profile 401 standard-EDS** in the Vendor field, ● **401** in the Profile field, ● click on **Add**, ● Enter 127 in the field **Node ID** then enter a name of a slave device in the **Description** field, **Result**: 
4	confirm by clicking **Ok**.

How to configure slave 8 The table below shows the different stages to configure the slave 8.

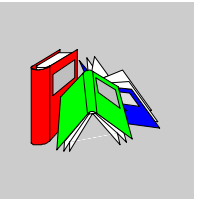
Step	Action
1	Double click on slave 8. **Result:** The following screen appears: 
2	Uncheck the **Activate node in actual configuration** box.
3	Uncheck the box **Automatic COB-ID allocation in accordance with Profile 301**.
4	Click on **Define new Transmit PDO** . **Result**: the following screen appears: 

Step	Action
5	Enter in a name for the PDO and confirm by clicking **Ok**.
6	Enter in a symbolic name (column **Symbolic Name**). Enter in COB-ID, 520 for example. Caution, this number should be the same as that of slave 8 on bus A. **Result**: 
7	Click on **PDO Characteristics**. **Result:** The following window appears: 
8	• Select **transmission event of transmit PDO defined in the device profile of the node**, • Select **no remote request, transmission of transmit PDO fully node dependent**, • confirm by clicking **Ok**.

Step	Action
9	Select **PDO_1_Dummy** in the **Configured PDOs** field then click on the key **PDO Contents Mapping**. **Result:** The following window appears: 
10	Select **Write_8_Inputs_1H-8H (Obj.Idx. 6200, Sub.Idx. 1)** (use scroll bar), click on **Append Object** then confirm by clicking **Ok**.

Step	Action
11	Confirm configuration screen by clicking on **Ok**. **Result**: The following structure appears: 

Glossary



C

CAN — **Controller Area Network**: field bus originally developed for automobile applications which is now used in many sectors, from industrial to tertiary.

CiA — **CAN in Automation**: international organization of users and manufacturers of CAN products.

COB — **Communication OBject**: transport unit on CAN bus. A COB is identified by a unique identifier, which is coded on 11 bits, [0, 2047]. A COB contains a maximum of 8 data bytes. The priority of a COB transmission is shown by its identifier - the weaker the identifier, the more priority the associated COB has.

CRC — **Cyclic Redundancy Checksum**: cyclic redundancy checksum indicates that no character has been "deformed" during frame transmission.

CSMA/CA — **Carrier Sense, Multiple Access / Collision Avoidance**: communication control method on a network featuring the link layer.

D

DIN — **Deutsches Institut für Normung**: German standardization institute.

DS — **Draft Standard**: specifications document created by the CIA organization.

E

EDS — **Electronic Data Sheet**: description file for each CAN device (provided by the manufacturers). To add a CAN device to the bus with Sycon configuration software, select the corresponding EDS. The EDS are available on the website **http://www.can-cia.de** or from the hardware provider.

L

Life Time — **Life Time = Life Time factor x Guard Time**.

LLC — **Logical Link Control**.

M

MAC — **Medium Access Control**.

MDI — **Medium Dependent Interface**.

MTBF — **Mean Time Between Failure**: mean time between two failures.

O

OD — **Object Dictionary**: object dictionary recognized by CAN. A hexadecimal code is given to each objet type, the dictionary regroups all the object's codes.

P

PCMCIA — **Personal Computer Memory Card International Association**

PDO **Process Data Object**: there are RPDOs (Receive PDO) and TPDOs (Transmit PDO).

PDU **Process Data Unit**: there are APDUs (Application PDUs). A PDU on the link layer is an APDU encapsulated by headings and bytes, which characterize this link layer.

PMA **Physical Medium Attachment**.

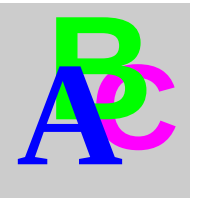
S

SDO **Service Data Object**: there are SSDOs (Server SDO) and CSDOs (Client SDO).

T

TAP **Transmission Access Point**: the bus connection unit.

Index



A

Accessing the Configuration Screen, 38
Architecture, 11

B

Bus length, 12

C

CANopen, 10
Characteristics, 26
Configuration, 37
  CANopen Bus Behavior, 41
  How to, 50
  Loading, 44
  Slave Inputs, 41
  Slave Outputs, 41
Configuration Screen, 39

D

Debugging, 68
diagnostic
  command, 66
Diagnostics, 70, 72

E

Error codes, 88
Exchange management, 85
Explicit exchange, 84

F

Folder, 53

G

General, 9

I

IDENTIFICATION, 63
Implicit exchange, 80
Input Words, 81
Installation
  Principle, 33

L

Language Objects
  Implicit, 80
Language objects, 79
  Configuration, 86
  Explicit, 84
LEDs, 71

M

Methodology, 35
Mounting the PCMCIA Card, 19, 21

O

Output words, 83

P

PDU, 61
Physical Description CPP 100/110, 17
Processors, 28

R

READ_VAR, 56
Report, 85

S

Select a configuration file, 50
Signaling, 71
Software, 31
Standards, 26

T

Topology, 11
Transmission speed, 12
TSX CPP 100/110, 15
TSX CPP ACC1, 23

W

WRITE_VAR, 56