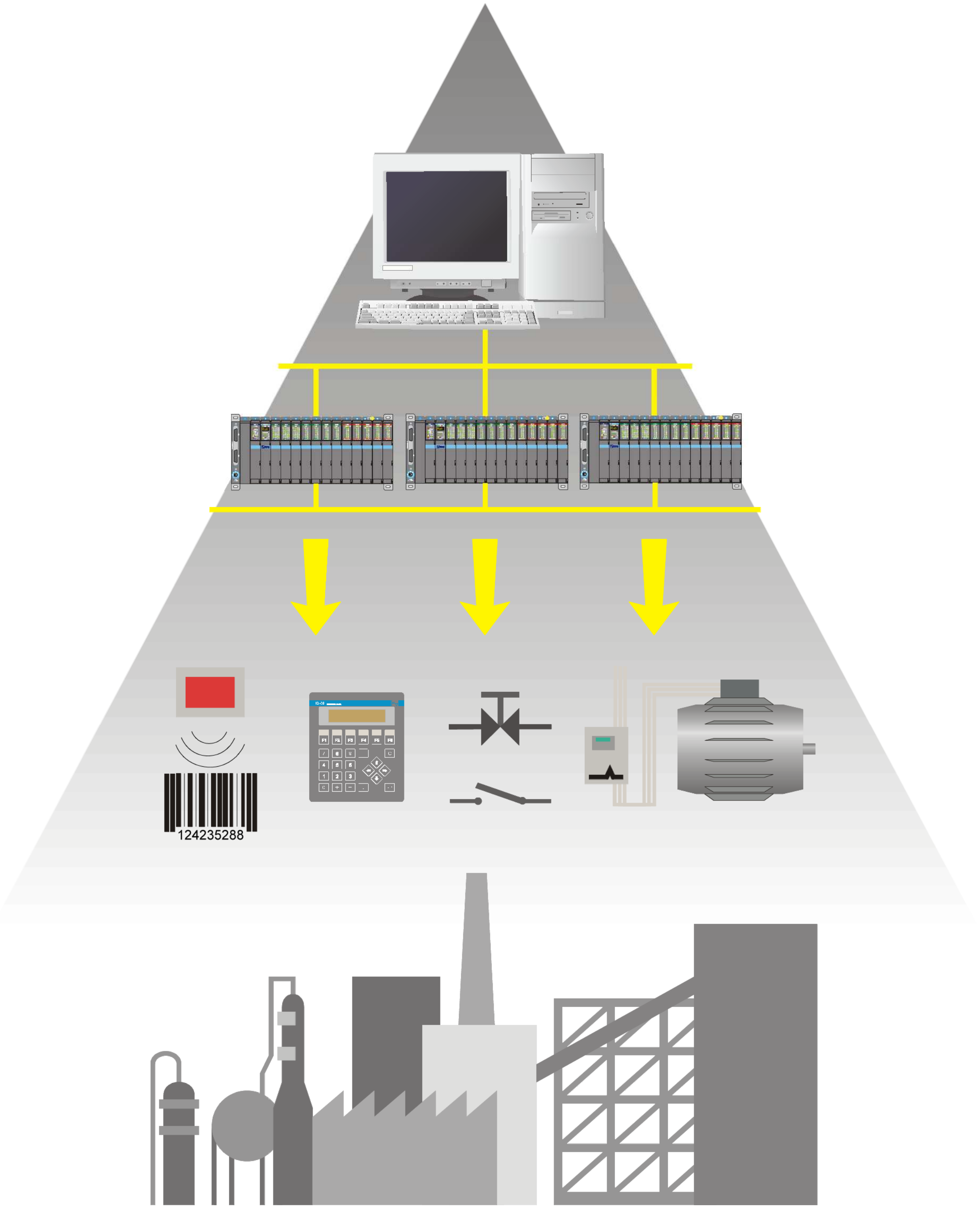
tecomat®

PROGRAMMABLE CONTROLLERS



SERIAL COMMUNICATION OF TECOMAT PROGRAMMABLE LOGIC CONTROLLERS – 32 BIT MODEL

SERIAL COMMUNICATION OF TECOMAT PROGRAMMABLE LOGIC CONTROLLERS 32 BIT MODEL

15th edition – March 2009

CONTENT

1. OVERVIEW OF TECOMAT SYSTEMS COMMUNICATION POSSIBILITIES

1.1. COMMUNICATION MODES OVERVIEW

The manual designation for individual systems

This manual is valid to full extent for the systems with the stack width of 32 bits only, i.e. programmable logic controllers TECOMAT TC650, TC700 and FOXTROT.

For the systems with the stack width of 16 bits, i.e. TECOMAT NS950, TC400, TC500, TC600 programmable logic controllers and freely programmable versions of TECOREG TR050, TR200 and TR300 controllers, there is the Serial communication of TECOMAT PLCs and TECOREG controllers - model 16 bits TXV 001 06.02 manual appointed. Further to this, the possibilities of serial communication of the TECOMAT programmable logic controllers differ in the type of system and the type of the central unit used.

Communication channels

There are three types of communication channels:

- serial channels CHx – are equipped with removable interface that enables high usage variability
- Ethernet ETHx interface
- USB interface

Serial channels

All central and communication units contain serial channels CH1, CH2, CH3, etc. Their features are described in the chapter 4.1. Serial channels can operate in following modes:

EIO mode - connection of other racks with peripheral modules (see chapter 2.1.)

PC mode - master system connection; usually PC or master system TECOMAT or TECOREG via EPSNET network; public and system services are available (see chapter 2.2.)

PLC mode - interconnection of more TECOMAT and TECOREG systems for the purpose of fast data forwarding (see chapter 2.3.)

MPC mode - slave systems network connection (EPSNET multimaster network) for the purpose of data interchange (see chapter 2.4.)

UNI mode - general user channel for universal use (connection of frequency converters, ID-04, ID-05, ID-07, ID-08 operator panels, bar code scanners, intelligent sensors, etc. - see chapter 2.5.)

MDB mode - connection of the master system; usually PC or master system of another producer via MODBUS network (see chapter 2.6.)

PFB mode - connection of PROFIBUS DP slave stations to the PLC (see chapter 2.7.)

UPD mode - operation of special submodules (see chapter 2.8.)

DPS mode - PROFIBUS DP slave station implementation (see chapter 2.9.)

CAN mode - connection of CANopen stations to the PLC (see chapter 2.10.)

CAS mode - CANopen station implementation (see chapter 2.11.)

CAB mode - CAN bus connection (see chapter 2.12.)

CSJ mode - CAN bus connection (see chapter 2.13.)

The availability of the above mentioned modes for individual serial channels is dependent partly on the type of the central unit used and partly on the type of the submodule fitted (see chapter 1.3.). Every removable module is identified and determines acceptable modes. If the user attempts to select a mode that is not supported by the removable submodule, the serial channel is switched off, meaning it switches to the **OFF** mode.

Case special is a central unit CP-7005 TC700 that uses serial channels CH1 and CH2 for system communication only. Channel CH1 is set to **SYN** mode and is used for synchronization of both parts of redundant system. Channel CH2 is set to **UPD** mode and is used for connection of the control panel of redundancy ID-20. The user is not able to affect these settings.

PC, **PLC** and **MPC** modes use EPSNET communication network. The description of this network, public functions included, is covered in chapter 3.

Ethernet interface

Some central units and communication modules contain Ethernet interface marked ETH1, ETH2, etc.. The possibilities of this connection are described in chapter 4.2. Ethernet interface can work in following modes:

PC mode	- master system connection; usually PC to Ethernet network via EPSNET UDP protocol; public and system services are available (see chapter 2.2.)
PLC mode	- interconnection of more TECOMAT systems through Ethernet network via EPSNET UDP protocol for the purpose of fast data forwarding (see chapter 2.3.)
UNI mode	- transmitting and receiving of arbitrary data through UDP and TCP protocols (see chapter 2.5.)
MDB mode	- connection of the master system; usually PC or master system of another producer via MODBUS network (see chapter 2.6.)

Ethernet interface can work in several modes at one time. **PC** and **MDB** modes are permanently active, others are optional. Ethernet interface can, therefore, be used also for programming and debugging using the Mosaic development environment.

Case special is central unit CP-7005 TC700 that uses Ethernet interface ETH1 for system communication only. ETH1 interface is set to **RED** mode and is used for synchronization of both parts of redundant system. The user is not able to affect these settings.

USB interface

Some central units have the USB interface. The possibilities of this connection are described in chapter 4.3. The data exchange itself via the EPSNET protocol corresponds to the **PC** mode, public as well as system functions are available thus, the USB interface is designed for programming and debugging using the Mosaic development environment.

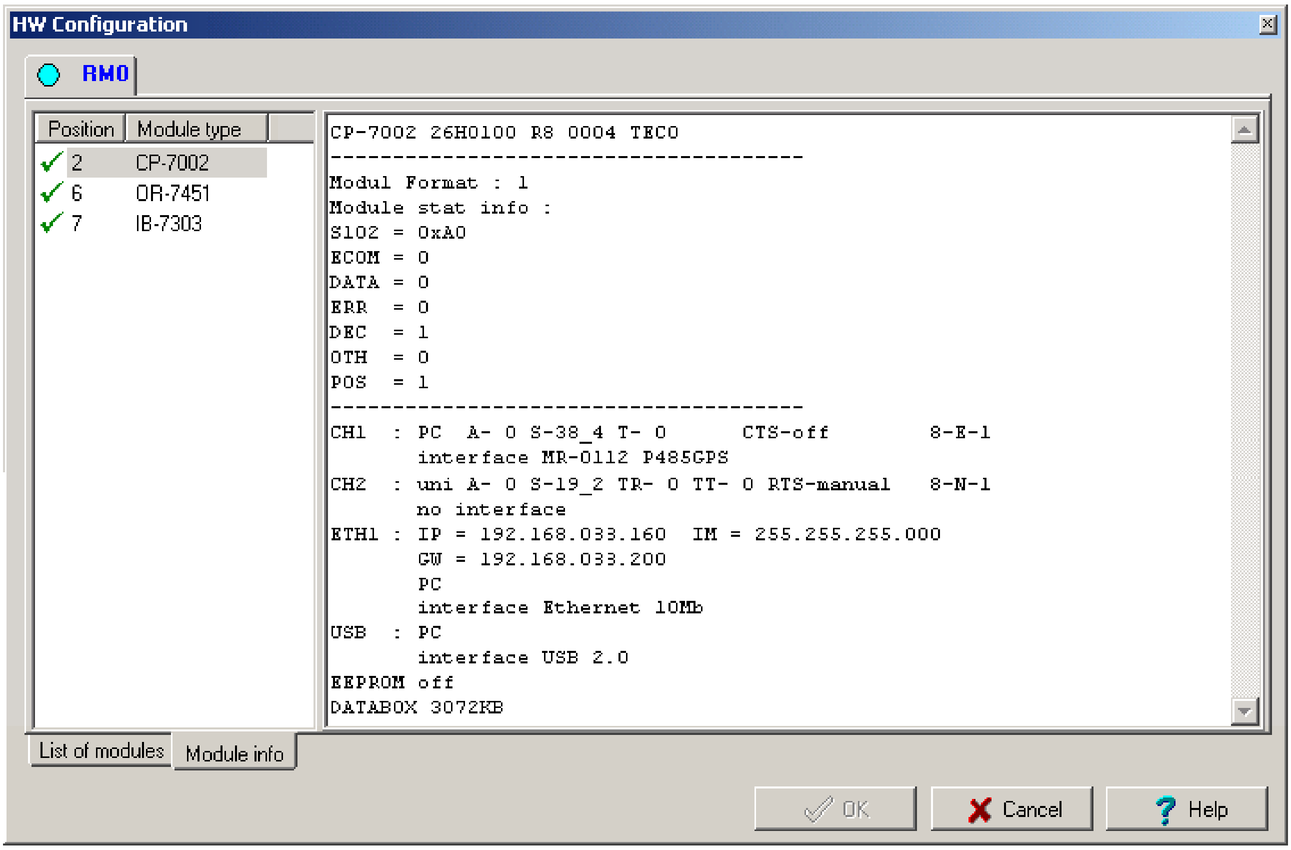
1.1.1. Communication parameter setup detection

In the Mosaic development environment, it is possible to find out the central unit setting by means of the option *PLC* | *HW configuration*, tag *Module info* (fig. 1.1).

On the first line of information you can find the identification of a concrete module, for example:

```
CP-7002 26H0100 R8 0004 TECO
```

where **CP-7002** is module type
26H0100 is identification of the sw version 2.6, version hw 01, revision hw 00
R8 0004 is the serial number of module
TECO is the name of the module producer



Position	Module type
✓ 2	CP-7002
✓ 6	OR-7451
✓ 7	IB-7303

```
CP-7002 26H0100 R8 0004 TECO
--------------------------------------
Modul Format : 1
Module stat info :
S102 = 0xA0
ECOM = 0
DATA = 0
ERR  = 0
DEC  = 1
OTH  = 0
POS  = 1
--------------------------------------
CH1  : PC  A- 0 S-38_4 T- 0      CTS-off        8-E-1
       interface MR-0112 P485GPS
CH2  : uni A- 0 S-19_2 TR- 0 TT- 0 RTS-manual   8-N-1
       no interface
ETH1 : IP = 192.168.033.160  IM = 255.255.255.000
       GW = 192.168.033.200
       PC
       interface Ethernet 10Mb
USB  : PC
       interface USB 2.0
EEPROM off
DATABOX 3072KB
```

Fig. 1.1 Central unit settings information

In the next section, the status of data exchange is given (a detailed description can be found in the relevant system manual, chapter diagnostics).

In the last section, there is an overview of current setup of communication channels and further central unit equipment. Each communication channel has its logic identification (CH1, CH2, etc.). Followed by a set mode, address (A), transmission rate (S - in kBd), response delay (T), transmission delay (P), maximum allowed space between received characters (B), idle condition among received messages (TR), idle condition among sent messages (TT), signal detection CTS, RTS signal mode and token mode (MT). At the end of the line is the format of the data being passed having the format of a-b-c, where:

a is the number of data bits (8 or 7)
b is the type of parity bit (E - even parity, D - uneven parity, N - no parity, 0 - parity 0, 1 - parity 1)
c is the number of stop bits (usually 1)

The type of plug-in submodule with interface is specified on the next line (if the submodule is not identified, the text „no interface“ is displayed).

Logic identification (ETH1 or ETH2), the IP address and the mask are used for the Ethernet interface. On the next line, active modes of communication and their possible settings are listed followed with transmission rate specification (10 Mb) on the next line.

Within the USB interface, communication mode and implemented interface version are given (currently 2.0).

Individual communication modules SC-7101 and SC-7102 release setting of their serial channels and the Ethernet interface separately in the same format as the central unit.

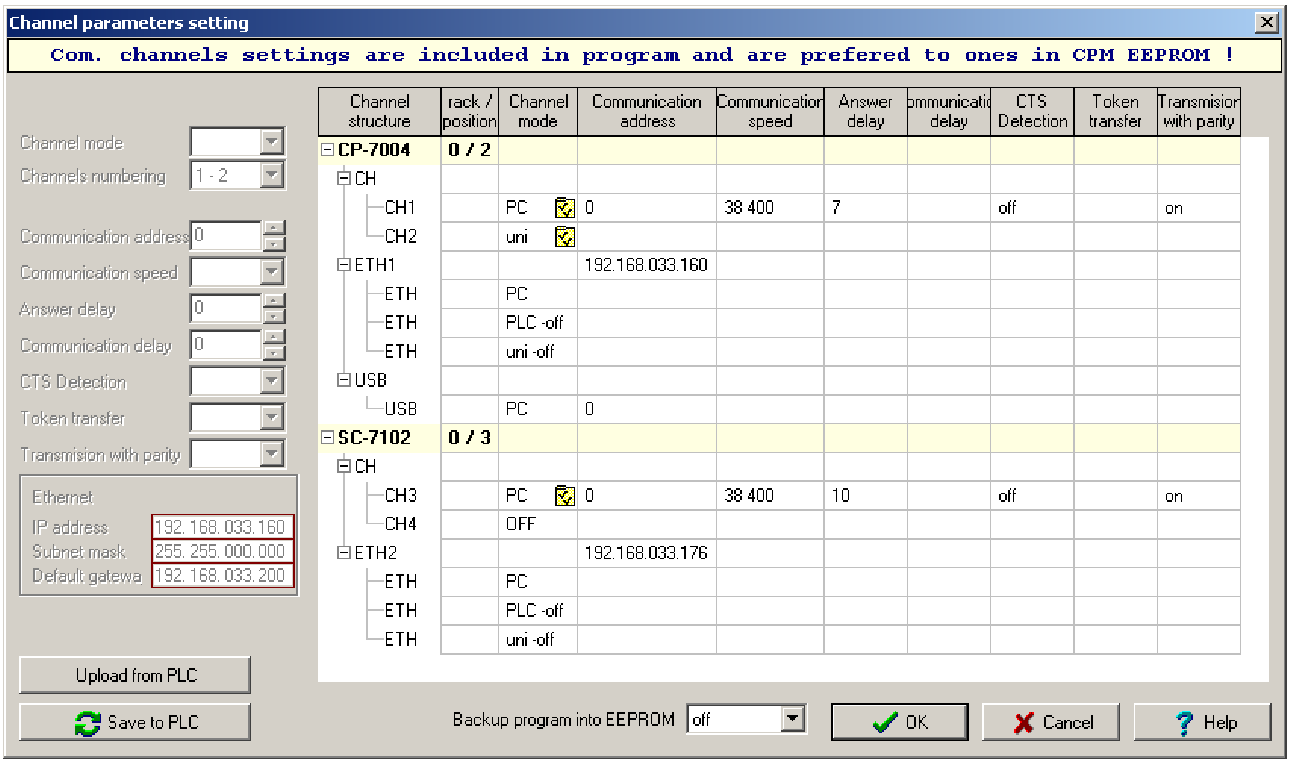
1.1.2. Communication parameters setup

Central units enable the setup of communication parameters at the first two serial channels (CH1 and CH2) and at the Ethernet ETH1 interface using push-buttons or the Mosaic development environment. All communication channels can be then set within the project. The necessary information is transferred to the user program that sets during PLC restart the required communication mode to the corresponding channel. Thus all the communication channels are set, including those being realized via SC-710x TC700 communication modules and which can not be modified using push-buttons.

If the communication channel is set to one of the modes using push-buttons and there is another setting in the user program then, at the moment of PLC restart the channel will be re-set according to data from the user program. Therefore, if we want any of the communication channels at the central unit to be adjustable independently of the user program, we must switch this communication channel off (**OFF** mode) during the compilation.

In the Project manager we select folder *Hw* | *HW Configuration*. A table displays, listing PLC configuration. We select the required central unit and press push-button *Settings* or the icon ☑ on the line of the central unit. The *Channel parameters setting* panel (fig. 1.2) that enable setup of all central unit parameters, will appear.

By pressing the push-button *Load from PLC,* parameters setup will be loaded as it is stored in the central unit. By pressing the push-button *Save to PLC,* parameters are saved to the central unit. New parameters are accepted after the user program restart is undertaken. This push-button is active only when the PLC is in the HALT mode.



Channel parameters setting

Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

Channel mode
Channels numbering: 1 - 2
Communication address: 0
Communication speed
Answer delay: 0
Communication delay: 0
CTS Detection
Token transfer
Transmision with parity

Ethernet
IP address: 192. 168. 033. 160
Subnet mask: 255. 255. 000. 000
Default gatewa: 192. 168. 033. 200

Channel structure	rack / position	Channel mode	Communication address	Communicatior speed	Answer delay	ommunicatic delay	CTS Detection	Token transfer	Transmisior with parity
⊟CP-7004	0 / 2								
⊟CH									
CH1		PC ☑	0	38 400	7		off		on
CH2		uni ☑							
⊟ETH1			192.168.033.160						
ETH		PC							
ETH		PLC -off							
ETH		uni -off							
⊟USB									
USB		PC	0						
⊟SC-7102	0 / 3								
⊟CH									
CH3		PC ☑	0	38 400	10		off		on
CH4		OFF							
⊟ETH2			192.168.033.176						
ETH		PC							
ETH		PLC -off							
ETH		uni -off							

Backup program into EEPROM: off

Fig. 1.2 Communication channel parameters setup

Communication channels setup in this panel becomes part of the user program after the compilation. If we want the settings of communication channels in the user program to be different from the settings stored in the central unit, we must change the settings of the communication channels before compilation via the push-button *Save to PLC*.

1.2. COMMUNICATION POSSIBILITIES OF INDIVIDUAL PLC TYPES

1.2.1. TECOMAT TC650

TECOMAT TC650 is a compact programmable logic controller with binary and analog inputs and outputs and with an expansion opportunity. It contains the central unit of C range with 3 serial channels in line with actual configuration.

Details are presented in the *Technical equipment of programmable controllers TECOMAT TC650* manual, order No. TXV 138 22.01.

Communication options of compact PLCs are tabularly presented in fig.1.1. There is, in fig.1.2, an overview of communication modes supported by individual versions of central unit firmware. The required communication mode must be supported by both relevant communication channels and also by mounted exchangeable module (fig.1.9). Moreover, it must be supported by the central unit firmware.

Fig.1.1 Overview of communication possibilities of PLC TECOMAT TC650

	TC650
Number of serial channels Number of Ethernet interfaces	3 1
Serial channels:	
- number of channels	3
- available modes for CH1	PC+, PLC, UNI, MPC, MDB, PFB, UPD, DPS, CAN, CAS, CAB, CSJ
- available modes for CH2	EIO, PC+, PLC, UNI, MPC, MDB, PFB, UPD, DPS, CAN, CAS, CAB, CSJ
- available modes for CH3*	PC+, PLC, UNI, MPC, MDB, PFB
Ethernet interface:	
- number of channels	1
- available modes for ETH1	PC+, PLC, UNI

* If there is in the CH3 channel slot a submodule with analog channels mounted, then the channel CH3 is permanently switched off.

+ **PC mode** contains also system functions enabling programming and debugging of the system. Only one master system can be using system functions within one connection at one time.

Fig.1.2 Overview of communication modes supported by individual versions of central unit TC650 software (sw)

Mode	Channel	TC650
EIO	CH2	all versions
PC	CH1 - CH3 ETH1	all versions all versions
PLC	CH1 - CH3 ETH1	all versions all versions
UNI	CH1 - CH3 ETH1	all versions from 1.6 sw
MPC	CH1 - CH3	all versions
MDB	CH1 - CH3	all versions
PFB	CH1 - CH3	all versions
UPD	CH1, CH2	all versions
DPS	CH1, CH2	from 1.8 sw
CAN	CH1, CH2	all versions
CAS	CH1, CH2	all versions
CAB	CH1, CH2	from 1.1 sw
CSJ	CH1, CH2	from 2.0 sw

1.2.2. TECOMAT TC700

TECOMAT TC700 is a modular programmable logic controller designed for middle-sized applications. Up to 12 peripheral modules and the CP-700x central unit can be fitted in the rack. Further, the assembly can be expanded by connection of other racks mounted with other peripheral modules. Using SC-7101 and SC-7102 system communication modules, the number of serial channels and Ethernet networks can be increased. Communication modules SC-7102 can be mounted not only onto the central unit but also onto the expander SE-7132.

Details can be found in the *TECOMAT TC700 programmable logic controllers* manual, order No. TXV 004 02.01.

Communication possibilities of central units can be found in table 1.3. Communication possibilities of the expander can be found in table 1.4.

Table 1.5 shows the overview of communication modes supported by individual versions of central unit firmware and table 1.6 shows the overview of communication modes supported by individual versions of expanders and communication modules firmware.

The required communication mode must be supported by both the corresponding communication channels and the exchangeable submodule mounted and also must be supported by the corresponding central unit and communication module firmware.

Table 1.3 Overview of communication possibilities of PLC TECOMAT TC700

	CP-7001	CP-7002 CP-7003	CP-7004	CP-7005
Serial channels:				
- on the central unit - on the communication module SC-710x	2 2	2 8	2 8	0* 8
- available modes for CH1	PC+, PLC, UNI, MPC, MDB, UPD, DPS, CAN, CAS, CAB, CSJ	PC+, PLC, UNI, MPC, MDB, PFB, UPD, DPS, CAN, CAS, CAB, CSJ	PC+, PLC, UNI, MPC, MDB, PFB, UPD, DPS, CSJ	-
- available modes for CH2	PC+, PLC, UNI, MPC, MDB, UPD, DPS, CAN, CAS, CAB, CSJ	EIO, PC+, PLC, UNI, MPC, MDB, PFB, UPD, DPS, CAN, CAS, CAB, CSJ	EIO, PC+, PLC, UNI, MPC, MDB, PFB, UPD, DPS, CSJ	-
- available modes for CH3 - CH10	PC+, PLC, UNI, MPC, MDB, PFB	PC+, PLC, UNI, MPC, MDB, PFB	PC+, PLC, UNI, MPC, MDB, PFB	PC+, PLC, UNI, MPC, MDB, PFB
USB interface:				
- on the central unit	1	1	1	1
- available modes for USB	PC+	PC+	PC+	PC+
Ethernet interface:				
- on the central unit - on the communication module SC-7102	- 1	1 1	1 1	0* 1
- available modes for ETH1	-	PC+, PLC, UNI	PC+, PLC, UNI, MDB	-
- available modes for ETH2	PC+, PLC, UNI	PC+, PLC, UNI	PC+, PLC, UNI	PC+, PLC, UNI

* No. of interfaces available to the user. There are both serial channels and Ethernet interface pre-occupied on the central unit CP-7005 for the need of the system.

+ **PC mode** contains also system functions enabling programming and debugging of the system. Only one master system can be using system functions within one connection at one time.

Table 1.4 Overview of communication possibilities of PLC TECOMAT TC700 expanders

	SE-7132
Serial channels:	
- on the expander - on the communication module SC-7101	0* 8
- available modes for CH43 - CH4A	PLC, UNI, MPC, PFB

* Serial channels and Ethernet interface on the expander are pre-occupied for needs of the system.

Table 1.5 Overview of communication modes supported by indiv. versions of central units software (sw) and hardware (hw)

Mode	Channel	CP-7001	CP-7002 CP-7003	CP-7004	CP-7005
EIO	CH2	-	from 2.5 sw	-	-
PC	CH1, CH2	all versions	all versions	all versions	-
	CH3 - CH10	from 4.6 sw*	from 4.6 sw*	all versions	from 4.6 sw*
	USB	all versions	all versions	all versions	all versions
	ETH1	all versions	all versions	all versions	-
	ETH2	from 4.6 sw*	from 4.6 sw*	all versions	from 4.6 sw*
PLC	CH1, CH2	all versions	all versions	all versions	-
	CH3 - CH10	from 1.7 sw	from 1.7 sw	all versions	all versions
	ETH1	from 3.2 sw	from 3.2 sw	all versions	-
	ETH2	from 3.2 sw	from 3.2 sw	all versions	all versions
UNI	CH1, CH2	all versions	all versions	all versions	-
	CH3 - CH10	from 1.7 sw	from 1.7 sw	all versions	all versions
	ETH1	from 4.5 sw	from 4.5 sw	all versions	-
	ETH2	from 4.7 sw	from 4.7 sw	all versions	all versions
MPC	CH1, CH2	all versions	all versions	all versions	-
	CH3 - CH10	from 1.7 sw	from 1.7 sw	all versions	all versions
MDB	CH1, CH2	all versions	all versions	all versions	-
	CH3 - CH10	from 1.7 sw	from 1.7 sw	all versions	all versions
	ETH1	-	-	from 3.7 sw	-
PFB	CH1, CH2	-	from 2.5 sw	all versions	-
	CH3 - CH10	from 4.0 sw	from 4.0 sw	all versions	all versions
UPD	CH1, CH2	from 1.4 sw	from 1.4 sw	all versions	-
DPS	CH1, CH2	from 4.7sw a 02hw	from 4.7sw a 02hw	from 3.5 sw	-
CAN	CH1, CH2	from 2.7 sw	from 2.7 sw	-	-
CAS	CH1, CH2	from 3.7 sw	from 3.7 sw	-	-
CAB	CH1, CH2	from 4.0 sw	from 4.0 sw	-	-
CSJ	CH1, CH2	from 5.4 sw	from 5.4 sw	all versions	-

* Older versions CP-7001, CP-7002, CP-7003, CP-7005, SC-7101 and SC-7102 also support **PC** mode**,** however, they are not compatible with the current version of data transmission that eliminated problems arising during the usage of more modules SC-710x at one time.

If not specified expressly, the support does not depend on hardware version. The latest software and hardware versions of the corresponding module can be found in the Mosaic development environment by selecting *PLC* | *HW configuration*, tag *Module information* (fig. 1.1).

Tab.1.6 Overview of communication modes supported by indiv. versions of software (sw) and hardware (hw) of expanders and communication modules

Mode	Channel	SE-7132	SC-7101 SC-7102
PC	CH3 - CH10 ETH2	- -	from 3.1 sw* from 3.1 sw*

PLC	CH3 - CH10
	CH43 - CH4A
	ETH2
		-
		from 1.5 sw
		-
			from 2.0 sw
			from 2.0 sw
			from 2.4 sw

UNI	CH3 - CH10
	CH43 - CH4A
	ETH2
		-
		all versions
		-
			from 2.0 sw
			from 2.0 sw
			from 3.2 sw

MPC	CH3 - CH10 CH43 - CH4A	- from 1.5 sw	from 2.0 sw from 2.0 sw
MDB	CH3 - CH10	-	from 2.0 sw
PFB	CH3 - CH10 CH43 - CH4A	- from 1.2 sw	from 2.6 sw from 2.6 sw

* Older versions CP-7001, CP-7002, CP-7003, CP-7005, SC-7101 and SC-7102 also support **PC** mode, however, they are not compatible with the current version of data transmission that eliminated problems arising during the usage of more modules SC-710x at one time.

1.2.3. TECOMAT FOXTROT

TECOMAT FOXTROT is a compact programmable logic controller containing binary and analog inputs and outputs with the possibility of extension. It contains the G range central unit with 2 serial channels.

Details are referred to in the Programmable controllers TECOMAT FOXTROT manual, order No. TXV 004 10.01.

Communication possibilities of individual central units can be found in the table 1.7.

An overview of communication modes supported by individual versions of central units firmware is shown in the table 1.8.

The required communication mode must be supported by both the corresponding communication channels and the exchangeable submodule mounted (table 1.9), and also must be supported by the corresponding central unit firmware.

Table 1.7 Overview of communication possibilities of PLC TECOMAT FOXTROT

	CP-1004, CP-1005 CP-1014, CP-1015
Serial channels:	
- on the central unit	2
- available modes for CH1	PC+, PLC, UNI, MPC, MDB, PFB
- available modes for CH2	PC+, PLC, UNI, MPC, MDB, PFB, UPD, DPS, CSJ
Ethernet interface:	
- on the central unit	1
- available modes for ETH1	PC+, PLC, UNI, MDB

+ **PC** mode contains also system functions enabling programming and debugging of the system. Only one master system can be using system functions within one connection at one time.

Tab.1.8 Overview of communication modes supported by individual versions of software (sw) and hardware (hw) of central units

Mode	Channel	CP-1004, CP-1005 CP-1014, CP-1015
PC	CH1, CH2 ETH1	all versions all versions
PLC	CH1, CH2 ETH1	all versions all versions
UNI	CH1, CH2 ETH1	all versions all versions
MPC	CH1, CH2	all versions
MDB	CH1, CH2 ETH1	all versions from 3.7 sw
PFB	CH1, CH2	all versions
UPD	CH2	all versions
DPS	CH2	from 3.5 sw
CSJ	CH2	all versions

If not specified expressly, the support does not depend on hardware version. The latest software and hardware versions of the corresponding module can be found in the Mosaic development environment by selecting *PLC* | *HW configuration*, tag *Module information* (fig. 1.1).

1.3. THE USE OF EXCHANGEABLE SUBMODULES WITHIN INDIVIDUAL MODES

Exchangeable modules are always intended for a certain group of communication modes. If there is a mode set on the communication channel that is not supported by a submodule then this channel is automatically switched off (set to **OFF** mode). Communication modes supported by individual exchangeable submodules are listed in the table 1.9. In the table 1.10 there is shown for which modules particular submodules can be used.

Table 1.9 Overview of communication modes supported by exchangeable submodules

Type	Function	Order No.	Supported modes
MR-0104 MR-0114 MR-0124	serial interface RS-232 serial interface RS-485 serial interface RS-422	TXN 101 04 TXN 101 14 TXN 101 24	PC, PLC, UNI, MPC, MDB, PFB
MR-0151	sequencer CAN (I82527)	TXN 101 51	CAN, CAS, CAB
MR-0152	station PROFIBUS DP slave	TXN 101 52	DPS
MR-0154	peripheral modules connection TC700	TXN 101 54	EIO
MR-0155 MR-0156	modem FSK and a hub modem FSK, hub and parallel amplifier	TXN 101 55 TXN 101 56	UNI
MR-0157	peripheral modules connection TC700	TXN 101 57	EIO
MR-0158	interface M-Bus	TXN 101 58	UNI
MR-0160 MR-0161	sequencer pair CAN (SJA1000) sequencer CAN (SJA1000)	TXN 101 60 TXN 101 61	CSJ
MX-0301	bar-code leader connection	TXN 103 01	UPD

Table 1.10 Overview of submodules supported by individual modules

Type	Remark	TC650	TC700 CP-7001 CP-7002 CP-7003	TC700 CP-7004	TC700 SC-7101 SC-7102	FOXTROT CP-1004 CP-1005 CP-1014 CP-1015
MR-0104 MR-0114 MR-0124	RS-232 RS-485 RS-422	x x x	x x x	x x x	x x x	x x x
MR-0151	CAN	x	x			
MR-0152	PROFIBUS	x	x	x		x
MR-0154	EIO	x	x			
MR-0155 MR-0156	FSK modem FSK modem	x x	x x	x x	x x	
MR-0157	EIO			x		
MR-0158	M-Bus	x	x	x	x	x
MR-0160 MR-0161	2x CAN CAN	x x	x x	x x		x x
MX-0301	bar-code	x	x	x		x
PX-7811 PX-7812	8x DI 4x DI, 4x DO	x x	x x	x x		x x

2. COMMUNICATION MODES

There are, in the following chapters, described individual communication modes. Information introduced here are valid for all interfaces for which is a particular mode accessible (serial channels, Ethernet, USB).

Table 2.1 Overview of communication modes and their accessibility for individual interfaces

Communication mode	Function	Interface	Descript.
EIO	peripheral modules connection	serial channel	chapter 2.1.
PC	communication with master system via EPSNET protocol	serial channel, Ethernet, USB	chapter 2.2.
PLC	data sharing network	serial channel, Ethernet	chapter 2.3.
MPC	data transmission among slave and master systems	serial channel	chapter 2.4.
UNI	general user channel	serial channel, Ethernet	chapter 2.5.
MDB	communication with master system via MODBUS protocol	serial channel, Ethernet	chapter 2.6.
PFB	connection of stations PROFIBUS DP slave to PLC	serial channel	chapter 2.7.
UPD	special submodules control	serial channel	chapter 2.8.
DPS	station PROFIBUS DP slave realization	serial channel	chapter 2.9.
CAN	connection of stations CANopen to PLC	serial channel	chapter 2.10.
CAS	station CANopen realization	serial channel	chapter 2.11.
CAB	connection of bus CAN	serial channel	chapter 2.12.
CSJ	connection of bus CAN	serial channel	chapter 2.13.

2.1. EIO MODE - CONNECTION OF PERIPHERAL MODULES

The **EIO** mode, which allows the connection of other racks with TC700 peripheral modules above the scope of standard system composition, is contained in central units on selected channels (see chapter 1.2.).

2.1.1. Channel parameters setup

PLC TECOMAT TC700 uses for the operation of peripheral modules an internal bus. This bus enables the support of 4 racks RM-7942 in total, or more precisely up to 8 racks RM-7941. When a higher number of racks or remote peripheral modules must be operated then the CH2 channel in the **EIO** mode for central units CP-7002, CP-7003 and CP-7004 can be used. Concerning this solution, no changes to the user program or any special settings are required.

The compact PLC TECOMAT TC650 can be extended to up to 4 modules of TC700 range. The channel CH2 in the EIO mode is used for this purpose. At this solution, no changes to the user program or any special settings are required.

To connect the peripheral modules and the central unit via the CH2 channel, it is vital to fit this channel with a special submodule MR-0154 (TXN 101 54), or more precisely MR-0157 (TXN 101 57 – for CP-7004). If this submodule is already fitted, the CH2 channel is set to the **EIO** mode automatically and the peripheral modules connected to this line are operated in the same way as the peripheral modules on the internal bus.

Requirements and installation instructions are described in the TECOMAT TC700 programmable controllers manual, order No. TXV 004 02.01, or the Technical equipment of programmable controllers TECOMAT TC650 manual, order No. TXV 138 22.01.

2.1.2. Network operation

The TC700 peripheral system does not require any special setting. The only condition is to avoid collision of rack addresses and remote peripheral modules. Moreover, the shift of rack addresses must be taken in to consideration. The figure "4" must be added to the rack address connected via the CH2 channel. It means that racks with addresses 0, 1, 2, 3 set by the rotary switch are operated by the central unit as racks with addresses 4, 5, 6, 7.

Modules from TC700 range connected to the central unit TC650 does not require any special setting. The only requirement is that the rack where modules are fitted, must be set within address 0. Operated modules must be in 0, 1, 2 and 3 positions.

The communication itself is fully supported by the system and therefore, it is not necessary to deal with it in detail.

2.2. PC MODE - COMMUNICATION WITH MASTER SYSTEM

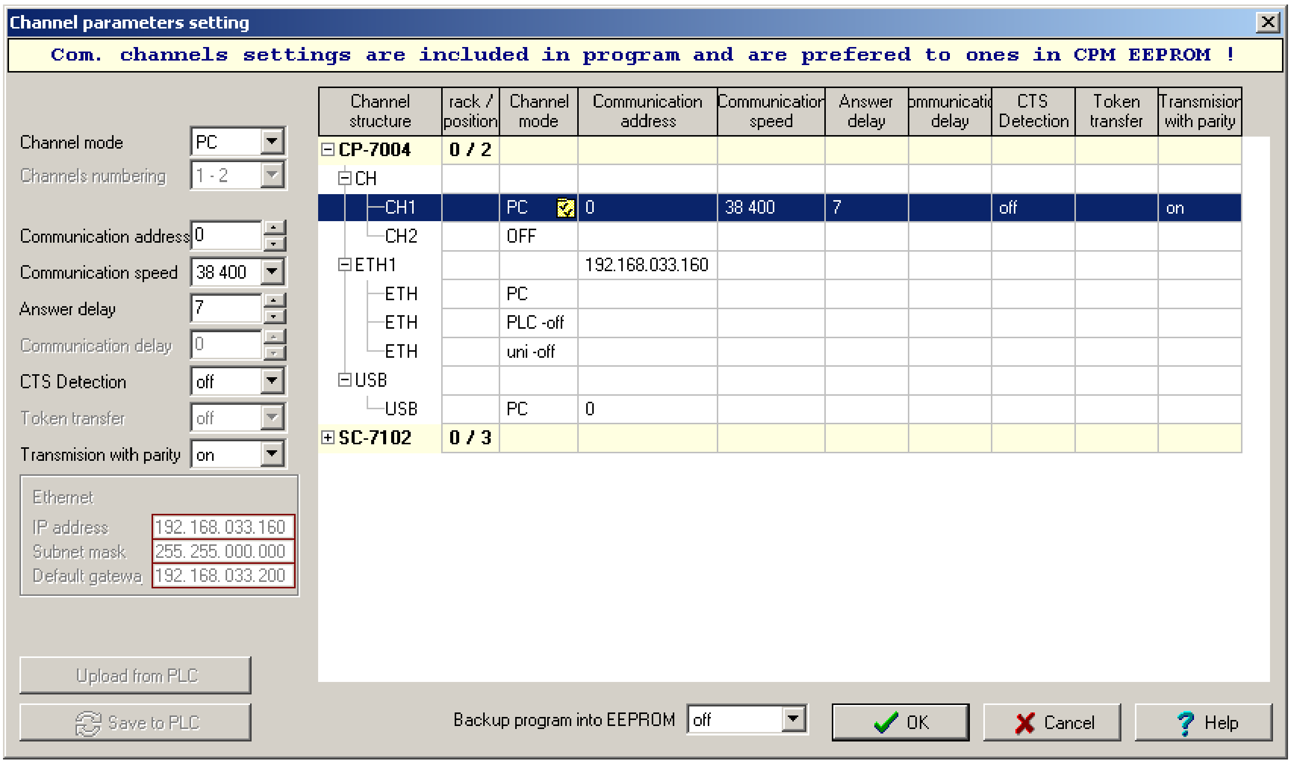
Communication is initiated by a master system on the query-response basis. This principle enables connection of wider number of participants to a master system on the RS-485 interface. Other interfaces enable connection of one participant (point to point connection). In the **PC** mode, the TECOMAT acts as a passive slave participant.

2.2.1. Channel parameters setup

In the Mosaic environment we select folder Hw | HW Configuration. In the list of modules we highlight, using the mouse, the central unit or the communication module of which we want the channel to be set. Then we press the push-button Settings or icon ☑ on the line to the right from the position number. Panel *Channel parameters setup* is displayed (fig. 2.2.1).

We set the relevant serial channel to the **PC** mode, then we set transmission rate, address, response timeout, CTS signal detection and parity. If we need to set the space between received characters, by pressing the ☑ icon on the line of the serial channel we open the panel of detailed channel setup (fig.2.2.2).

After pressing the push-button *Save to PLC,* settings will be saved to the central unit (this does not apply to serial channels of communication modules). If we undertake the user program compilation then the setting of communications will become a part of the user program and will be activated in the central unit right during the restart of user program. Furthermore, for communication modules there must be set the *Channel numbering* and so assigned a logical identification to serial channels (e.g. CH3, CH4). The central unit has its fixed assigned serial channels CH1 and CH2.



Channel parameters setting

Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

Channel mode: PC
Channels numbering: 1 - 2
Communication address: 0
Communication speed: 38 400
Answer delay: 7
Communication delay: 0
CTS Detection: off
Token transfer: off
Transmision with parity: on

Ethernet
IP address: 192. 168. 033. 160
Subnet mask: 255. 255. 000. 000
Default gatewa: 192. 168. 033. 200

Channel structure	rack / position	Channel mode	Communication address	Communicatior speed	Answer delay	ommunicatic delay	CTS Detection	Token transfer	Transmisior with parity
⊟CP-7004	0 / 2								
⊟CH									
CH1		PC ☑	0	38 400	7		off		on
CH2		OFF							
⊟ETH1			192.168.033.160						
ETH		PC							
ETH		PLC -off							
ETH		uni -off							
⊟USB									
USB		PC	0						
⊞SC-7102	0 / 3								

Backup program into EEPROM: off

Fig.2.2.1 **PC** mode setting

For the Ethernet interface the IP address, IP mask and eventually IP address of the gate (gateway) must be selected and saved to PLC. There is no setting required for the USB interface.

If the central unit CP-7001, CP-7002, CP-7003, CP-7005 TC700 uses the version sw 4.6 and higher or if it is the CP-7004 central unit then the communication module SC-710x must use the version 3.1 and higher owing to compatibility of data transmission. The change of data transmission was implemented due to prevention of collision among several communication modules together.

Address setting

The address selection enables the connection of more TECOMAT and TECOREG systems to one master system (in this case, the RS-485 interface is necessary). It must be ensured that connected PLCs as well as the master system always have a different addresses. The addresses do not need to constitute a continuous series.

Transmission rate setting

Serial channels in this mode enable the transmission rate of up to 115.2 kBd. The higher the transmission rate, the shorter the data transmission. Since the communication runs independently of the user program, it affects all the cycle phases of the user program evenly. It applies here that by increasing the transmission rate, the cycle time extends slightly. In the table 2.2.1 the cycle time extension in per cents is shown. These values are valid in case of the maximum channel load (continuous communication). The longer timeouts between the individual messages, the lower cycle time extension is.

Table.2.2.1 Average user program cycle time extension depending on the serial channel transmission rate and central unit type

Transmission rate	TC650, CP-7001, CP-7002, CP-7003, CP-7005 TC700	FOXTROT, CP-7004 TC700
9.6 kBd	0.3 %	0.1 %
19.2 kBd	0.5 %	0.1 %
38.4 kBd	1.0 %	0.2 %
57.6 kBd	1.5 %	0.3 %
115.2 kBd	3.0 %	0.6 %

Response timeout setting

Response timeout setting allows selection of the minimum time period that elapses between the sending of the last master system message byte and the start of the transmission of the first response byte of the inquired TECOMAT system. During this period, the master system must get ready for data acceptance. Sometimes, this preparation can take more time depending on the type of master system (e.g. on the operation system used on the PC). The use of some serial interface converters, repeaters, modems and radio modems also requires a longer response timeout. Usually, it is a device switching the direction of communication (e.g. to the RS-485 interface) or causing a long transfer time delay.

A minimum value of response timeout is set. Its maximum value, in majority of functions, depends on the cycle period of the TECOMAT system because the data is passed on only during the I/O scan to ensure their constancy while user program is performing. The minimum timeout response can be set fixly in the range of 1 to 99 ms or it can be set to 0 which signify that the minimum timeout response corresponds to the time needed for sending one byte and thus, it depends on the transmission rate according to the Table 2.2.2.

Table 2.2.2 Minimum response timeout at set value 0

Transmission rate	Minimum response timeout
0.3 kBd	36.67 ms
0.6 kBd	18.33 ms
1.2 kBd	9.17 ms
2.4 kBd	4.58 ms
4.8 kBd	2.29 ms
9.6 kBd	1.14 ms
14.4 kBd	0.76 ms
19.2 kBd	0.57 ms
28.8 kBd	0.38 ms
38.4 kBd	0.29 ms
57.6 kBd	0.19 ms
115.2 kBd	0.09 ms

CTS signal detection setup

Switching on the CTS signal detection enables to retain the external signal response from a modem. The response will be sent only after 10 ms following the CTS signal change to the corresponding RTS signal status valid for response sending.

CTS signal detection is primarily intended for the communication via modems. Even with the CTS signal detection, the minimum set timeout response is ensured at the same time. This means that TECOMAT will not start sending the response before the minimum response time elapses even if the CTS signal is already properly set.

The required solution to the advanced RTS setup before data broadcasting

A timeout of 10 ms after CTS change detection is intended to "calm" conditions within the transmission medium before data transmission starts (e.g. carrier frequency start). If the modem does not return the CTS signal but requires the 10 ms timeout between RTS signal setup and the data itself, we inter-connect the RTS and CTS signals on the serial channel connector and switch the CTS signal detection on. Setting of the minimum timeout response (previous parameter) does not guarantee the exact moment of RTS signal switching. It is switched only at the moment of response creation, here, the effect of cycle time approves itself significantly.

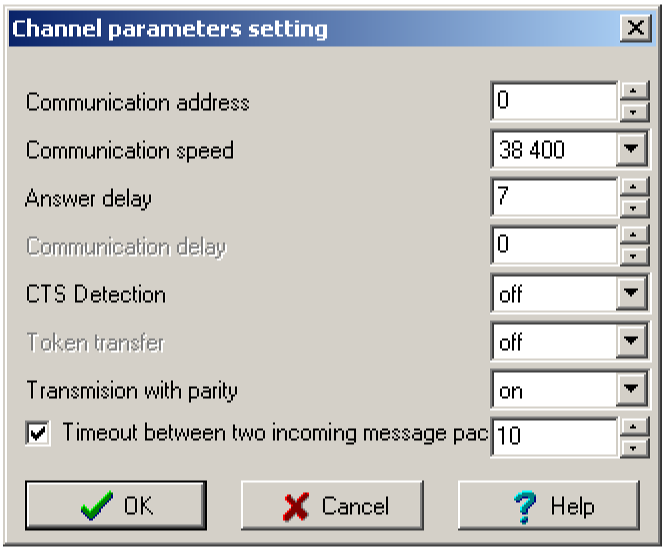
Parity setup

The EPSNET network protocol uses the even parity. However, some modems do not allow transmitting the parity. When used, the parity can be switched off on the serial channel of the central unit (in this case, the transmission without parity must be supported by the master system, too).

Attention! Parity considerably participates on the data transmission security. By its switch-off, the risk of undetected faulty data transmission is increased. Possibilities of additional data protection are introduced in chapter 3.

The space setup between received characters

The optional space between characters received is used for solution of events when the master system that sends the message out, or transmission device on the route (modems, serial interface converters), disturb the sending message in such a way that the maximum allowed space of 3 bytes between characters is not adhered. By setting this parameter to non-zero value, the PLC accepts a larger space in the middle of the receiving message up to the size set by the parameter.


Channel parameters setting

Communication address 0

Communication speed 38 400

Answer delay 7

Communication delay 0

CTS Detection off

Token transfer off

Transmision with parity on

☑ Timeout between two incoming message pac 10

Fig. 2.2.2 Detailed setting of ***PC mode***

Attention! As for security purposes, it is required that the heading of the message is received at full, i. e. the first 8 characters of the message must not be interrupted, only after then this parameter is accepted. This condition is usually fulfilled by modems due to balancing buffers.

The setup of this parameter is possible only within the detailed channel setting panel (fig.2.2.2, available via ☑ icon). By ticking the item *Timeout between two incoming message packets,* we unlock the setup of its size in the right box. The space value is set in ms within the range from 1 to 255 and it does not depend on a communication speed. The 0 value means that this function is switched off and PLC requires the maximum space of 3 bytes between characters to be adhered.

This parameter is supported by central units CP-7001, CP-7002, CP-7003, CP-7005 TC700 from version of sw 4.2, TC650 from version of sw 1.2, CP-7004 TC700 and FOXTROT of all versions.

2.2.2. Network operation

Public and system function of the EPSNET network

In the **PC** mode, the complete set of EPSNET network function is available within all communication channels, part of which are both system function used for system programming and debugging and public function used for data interchange. The limiting condition is that the system function are supported by one communication channel only at the given time.

The EPSNET network and its public function are described in chapter 3.

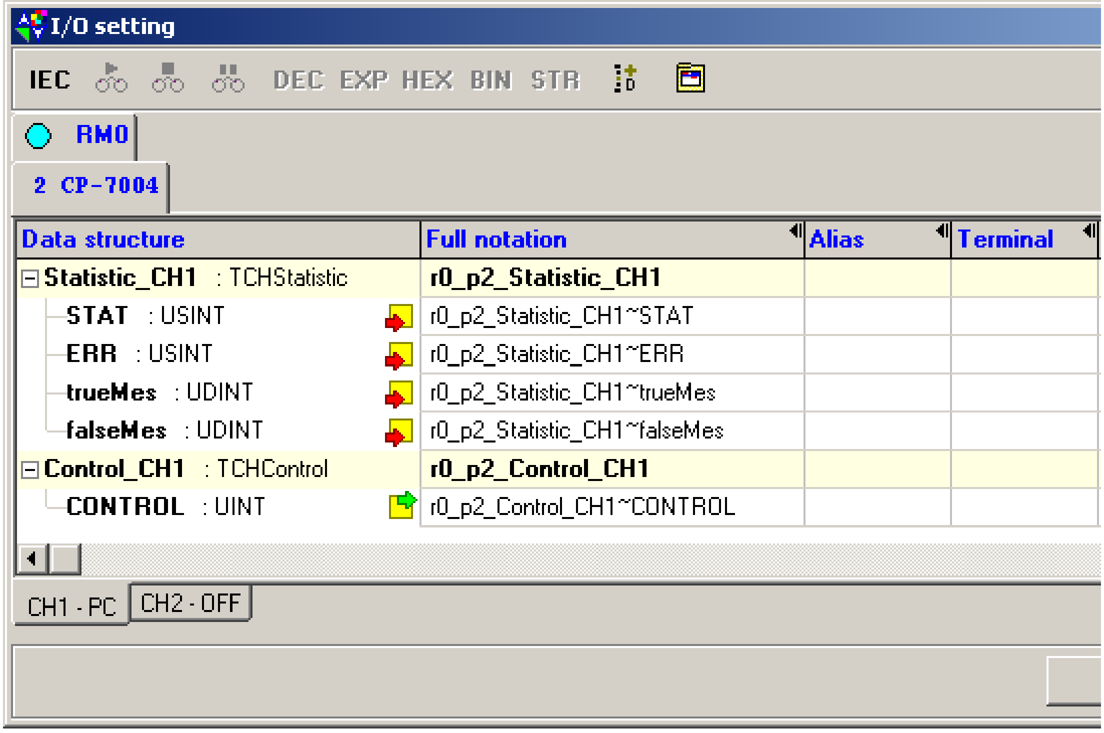
Warning: The Mosaic development environment uses EPSNET network system function and as a consequence, they can be operated through one communication channel only in the **PC** mode. The authorization for system function usage is given to the channel that sends the requirement as the first one. The authorization expires 5 seconds after the communication termination.

The operation diagnostics

Serial channels in the **PC** mode where their declaration is part of the user program, release the diagnostic data of the line status. These data are stored in the scratchpad and are easily accessible through the panel of *I/O setting*, accessed through the icon IO (fig. 2.2.3).

Diagnostic data have assigned symbolic names which begin with the rack and position number. In the column *Full notation*, a concrete symbolic name for the given item is always specified.

If we want to use data in the user program, we use either this symbolic name or we enter our own symbolic name in the column *Alias* which we can then use. In no case do we use absolute operands as they can alter after a new compilation of the user program.



Data structure	Full notation	Alias	Terminal
⊟ Statistic_CH1 : TCHStatistic	r0_p2_Statistic_CH1		
STAT : USINT	r0_p2_Statistic_CH1~STAT		
ERR : USINT	r0_p2_Statistic_CH1~ERR		
trueMes : UDINT	r0_p2_Statistic_CH1~trueMes		
falseMes : UDINT	r0_p2_Statistic_CH1~falseMes		
⊟ Control_CH1 : TCHControl	r0_p2_Control_CH1		
CONTROL : UINT	r0_p2_Control_CH1~CONTROL		

*Fig.2.2.3   Serial channel data in the **PC mode***

Network diagnostics:

Input data *rx_py_Statistic_XXX* (*TCHStatistic* structure):

(x -rack number, y -module position number in the rack, XXX -communication channel designation)

Stat	- communication status (not used for now) (usint type)
Err	- communication error (see chapter 13.6.) (usint type)
trueMes	- number of valid network communications (udint type)
falseMes	- number of invalid network communications (udint type)

Output data *rx_py_Control_XXX* (*TCHControl* structure):

(x -rack number, y -module position number in the rack, XXX -communication channel designation)

Control	- communication control (not used for now) (uint type)

Data exchanged with master participants:
Input data *Zone1In4* and *Zone2In4* (usint array):

user data received from one of master stations

Output data *Zone1Out4* and *Zone2Out4* (usint array):

user data sent to one of master stations

These zones exist only when this serial channel is connected to the PLC network where at least one of them stands for a master station (channel mode **MPC** - chapter 2.4.) and the communication with this station is programmed. Names of these zones are then assessed by symbolic naming in the table *Network setting* (fig. 2.4.3) that is relevant to the master station.

2.3. PLC MODE - DATA SHARING NETWORK

The communication channel in this mode is used for the interconnection of TECOMAT and TECOREG systems to ensure fast mutual data interchange. The communication is realized via EPSNET-F protocol within RS-485 serial interface that enables the network creation. **PLC** mode can be operated within Ethernet interface, too, via EPSNET UDP protocol.

There are, in all systems, passing zones reserved for each network user within same registers (data passing zone from one system is always located on same registers in all other systems interconnected to the network). In exchange to this express restriction, there is a high network throughput ensured because the identical location of passing zones in all systems allows the transmission of one data block to all network users simultaneously each time. Consequently, this causes considerable time saving and lower call on machine´s time of central unit processors.

The initialization of network passing zones is a part of user programs of interconnected systems and thus **must be identical for all systems** (if individual systems are part of the same group of projects then the request is ensured by Mosaic development environment. The relevant communication channel must be set to **PLC** mode and further the transmission speed and address is set.

PLC mode on the serial channel

The PLC mode is designed for fast data passing and no communication timeouts are taken into account on the serial channels. Therefore, **it is not possible to use modems or similar devices,** which delay data transmission on the line, while using this mode (the delay time longer than the time necessary to transmit ten bytes). If we have to use some of these devices, we interconnect the TECOMAT and TECOREG systems using the **MPC** mode where the transfer delay time can be set.

PLC mode on the Ethernet network

The **PLC** mode on the Ethernet network is realized in such a way that each participant sends data to the network using the broadcast type message (general circular - link address FF.FF.FF.FF.FF.FF) in a time cycle that can be set from 10 to 2550 ms range. Simultaneously, the participant receives these messages from other participants. The advantage of this type of communication is a high network throughput. But it ought to be remembered that broadcast type messages are received by all LAN users, even if they do not process them in any way. Moreover, the line is heavily loaded with this communication and consequently collisions may occur. Therefore, we do not recommend the **PLC** mode, intended for data exchange in real time, to be operated via the network used also by other participants which do not relate to the technology (standard company network). For data interchange among PLCs, it is more suitable to reserve an independent network.

2.3.1. Channel parameters setup

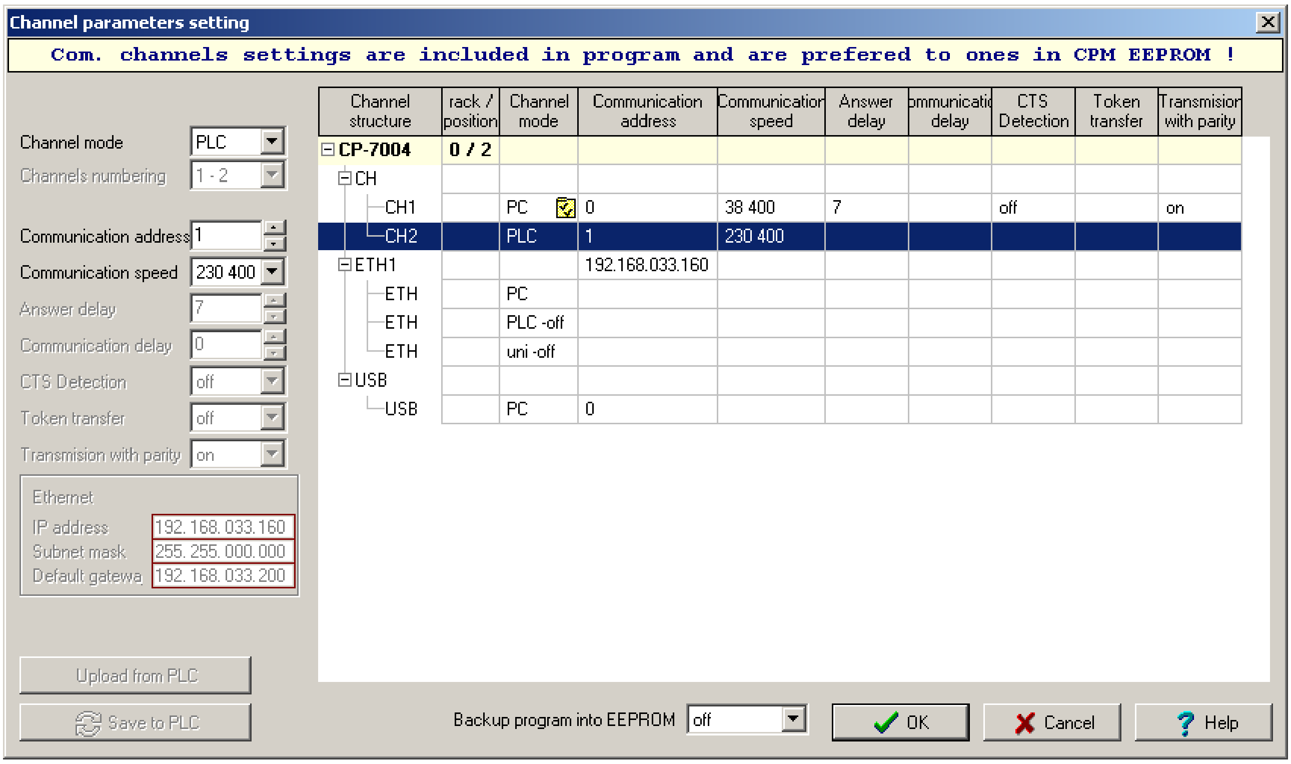
In the Mosaic development environment, we select folder *Hw* | *HW Configuration*. In the list of modules, we highlight (using the mouse) the central unit or the communication module, the channel of which we want to set. Then, we press the push-button Settings or icon ☑ on the line right from the position number. The *Channel parameters setting* panel (fig. 2.3.1) is displayed.

We set the corresponding serial channel to **PLC** mode and set the address and transmission rate. After compilation of the user program, the communication settings will become a part of the user program and will be activated in the central unit at the moment of restart of the user program.

It makes no sense to save the **PLC** mode in the central unit using the push-button *Save to PLC*, since this mode requires initialization data that are a part of the user program.

Moreover, for communication modules we must set the option *Channels numbering* and thus assign logic identification to serial channels (e.g. CH3, CH4). The central unit has the CH1 and CH2 serial channels fixed assigned.

For the Ethernet network, we must activate the **PLC** mode (item **PLC-off** is to be changed to **PLC**) and set the address. Furthermore, the IP address and the IP mask must be saved to the PLC.



Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

Channel mode: PLC
Channels numbering: 1 - 2
Communication address: 1
Communication speed: 230 400
Answer delay: 7
Communication delay: 0
CTS Detection: off
Token transfer: off
Transmision with parity: on

Ethernet
IP address: 192. 168. 033. 160
Subnet mask: 255. 255. 000. 000
Default gatewa: 192. 168. 033. 200

Channel structure	rack / position	Channel mode	Communication address	Communicatior speed	Answer delay	ommunicati delay	CTS Detection	Token transfer	Transmision with parity
⊟ **CP-7004**	**0 / 2**								
⊟ CH									
CH1		PC	0	38 400	7		off		on
CH2		PLC	1	230 400					
⊟ ETH1			192.168.033.160						
ETH		PC							
ETH		PLC -off							
ETH		uni -off							
⊟ USB									
USB		PC	0						

Backup program into EEPROM: off

Fig.2.3.1 ***PLC*** *mode setting*

Address setup

The address selection solves the resolution of individual TECOMAT and TECOREG systems on the network. Here, it is necessary to ensure that each of the connected systems has a different address. Addresses need not to form a continuous line. The addresses undertake the same resolution function also on the Ethernet network.

Transmission rate selection

The **PLC** mode enables the transmission rate on a serial channel of up to 230.4 kBd (according to the system type). The higher the transmission rate, the shorter data transmission time, however, the line’s resistance against disturbances decreases. We do recommend choosing only as high transmission rate that ensures and manages the data transfer in such time that we need considering the reaction requirements of the technology being controlled.

Data in passing zones located in the scratchpad are updated always during the I/O scan, so it is unnecessary e. g. to transmit the data three times during the system cycle. By decreasing the transmission rate, the resistance against interference increases and, in case of communication through serial channels of the central unit, cycle period shortening is reached, too.

In CP-700x TC700 and CP-100x FOXTROT central units, there is on the first two channels CH1 and CH2 and in TC650 central units the self-communication undertaken by the central unit processor. Regarding that the communication runs independently on the user program, it affects all parts of the user program cycle equally. Here applies that by increasing the transmission rate, the cycle time is extended. There is in fig.2.3.1 the extension of cycle period in per cents shown. Within the Ethernet network the transmission rate is not selected but it is set by network interface parameters.

Fig. 3.1 The average extension of user program cycle period depending on the transmission rate of the serial channel

Transmission rate	TC650, CP-7001, CP-7002, CP-7003, CP-7005 TC700*	FOXTROT, CP-7004 TC700*
9.6 kBd	0.3 %	0.1 %
19.2 kBd	0.5 %	0.1 %
38.4 kBd	1.0 %	0.2 %
57.6 kBd	1.5 %	0.3 %
115.2 kBd	3.0 %	0.6 %
172.8 kBd	4.5 %	0.9 %
230.4 kBd	6.0 %	1.2 %

* Communication channels of communication modules SC-7101, SC-7102 extend the cycle period only by use data transfer. The communication itself is controlled by the local processor of the communication module.

Application of communication modules SC-7101, SC-7102

If we use, for the central unit, the SC-7101 or SC-7102 communication module with other communication channels within the PLC TC700, it is then advisable to use one of these channels for the connection to the network. The communication load is then turned over to the processor of the communication module and it stops influencing the cycle period. The same is applicable for both serial channels and the Ethernet interface too.

2.3.2. Network initialization

Parameters setup in the Mosaic environment

The assignment of serial channel zones into the scratchpad in the Mosaic development environment is accomplished by filling in the form on basis of which the initialization of the channel is set automatically. This form is accessible from the project manager folder *Hw | PLC Network* (fig.2.3.2). On the screen we insert all PLCs from the project group, draw a bus and interconnect to it all channels in the **PLC** mode that we intend to have on one network. Using the right mouse click on any of channels in the **PLC** mode within this network the local menu will appear where we will select the *Network setting* (fig. 2.3.3) item.

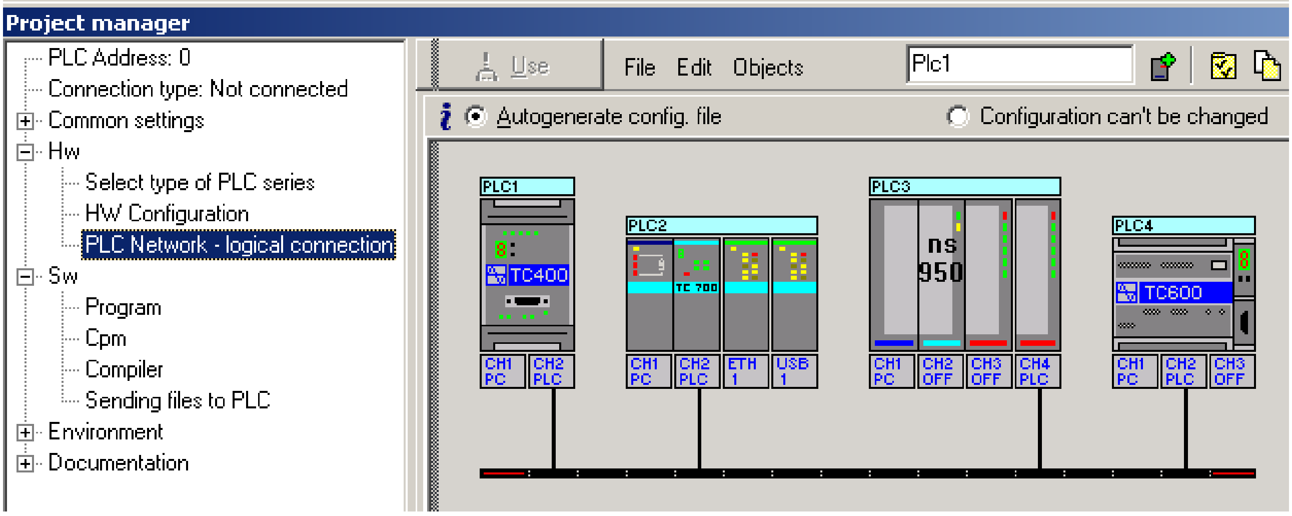
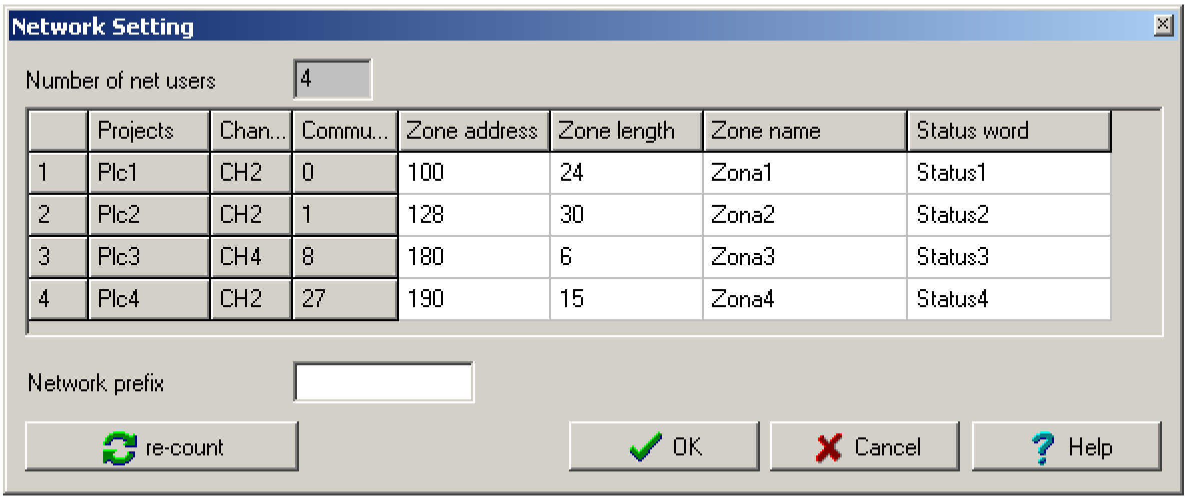


Fig.2.3.2 The PLC network creation according to an example


Network Setting

Number of net users: 4

	Projects	Chan...	Commu...	Zone address	Zone length	Zone name	Status word
1	Plc1	CH2	0	100	24	Zona1	Status1
2	Plc2	CH2	1	128	30	Zona2	Status2
3	Plc3	CH4	8	180	6	Zona3	Status3
4	Plc4	CH2	27	190	15	Zona4	Status4

Network prefix:

Fig.2.3.3 Network parameters setting according to an example

In the field *Network prefix* can be entered any network name which will be added as the prefix to all symbolic names of the network. Its use is of importance in case when more networks among the same systems are realized that must be differentiated from each other. On the Ethernet interface we proceed similarly (fig. 2.3.4). The only difference is that the Ethernet interface allows operation in more modes at the same time. Only such systems are included among the network participants that have the **PLC** mode activated.

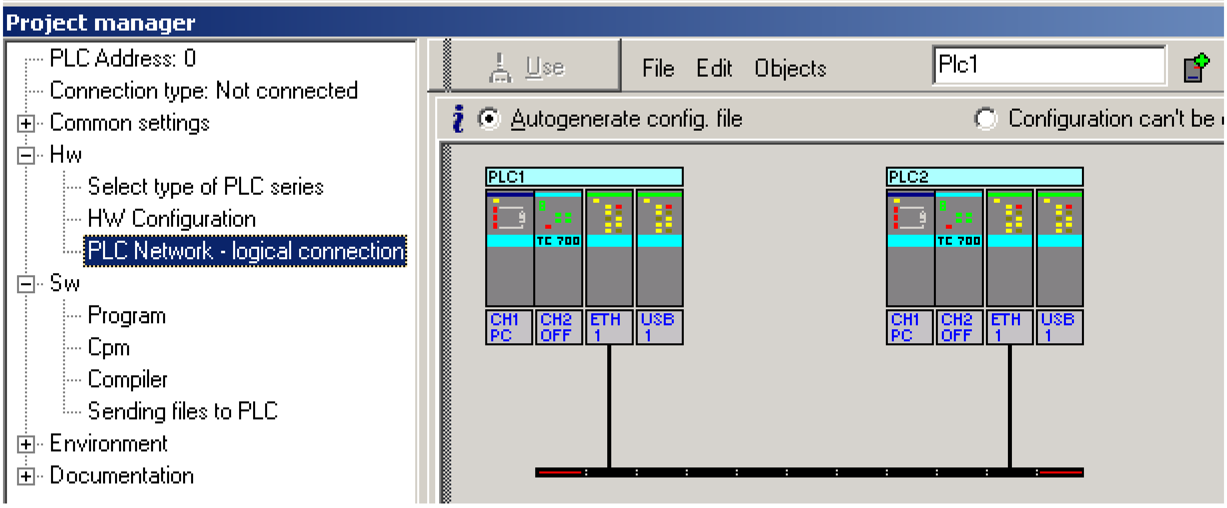


Fig.2.3.4 The PLC network creation on the Ethernet interface

Table.2.3.2 The maximum number of data transferred from one user depending on the total number of users

Total number of network users	Max. number of data transferred from one user TC650, CP-700x TC700, FOXTROT
2 up to 32	238 bytes

Note: The data number exceeding invoke the error 83 cc 4206 (cc stands for a number of serial channel used)

Example

Let us have a network of four systems, among which we want to transmit the following information:

1. TC400 addr. 0, transmission of 24 bytes, passing zone from register R100
2. TC700 addr. 1, transmission of 30 bytes, passing zone from register R128
3. NS950 addr. 8, transmission of 6 bytes, passing zone from register R180
4. TC600 addr. 27, transmission of 15 bytes, passing zone from register R190

In the Mosaic development environment, the table will be created automatically for all four systems that are part of the same project group according to fig. 2.3.2 and 2.3.3.

The distribution of passing zones in the scratchpad of each system will be as follows:

R0 R95	free for user program
R96 R99	system passing zone status addr. 0 *
R100 R123	system passing zone addr. 0
R124 R127	system passing zone addr. 1 *
R128 R157	system passing zone addr. 1
R158 R175	free for user program
R176 R179	system passing zone addr. 8 *
R180 R185	system passing zone addr. 8
R186 R189	system passing zone status addr. 27 *
R190 R204	system passing zone addr. 27

* The passing zone status is only in the systems with the 32 bit model. In 16 bit model systems, the status of the passing zone is not present (its function fulfils the network status zone), the space corresponding to the status size (4 bytes) is not used in these systems.

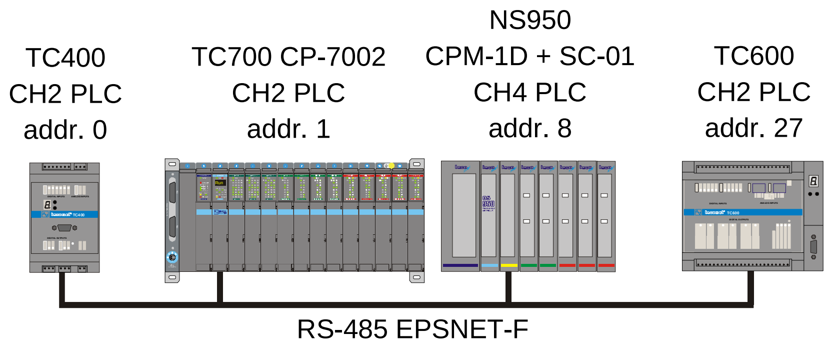


*Fig.2.3.5 Interconnection of network systems in **PLC** mode according to an example*

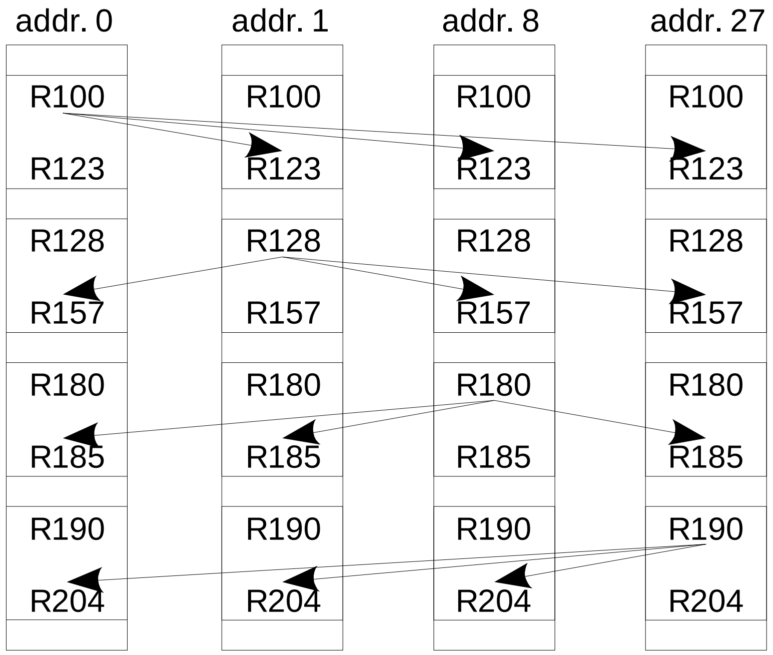


*Fig.2.3.6 Graphic visualisation of data interchange among systems in the **PLC** mode*

From the above mentioned it follows that passing zones need not to be necessarily connected continuously. However, generally we declare zones gathered. It is necessary though those zones including their statuses do not coincide with one another. The Mosaic environment supervise this error.

32 bits model systems have ensured that the transmission rate and address will be set after the restart of the user program according to parameters of the whole network generated by the compiler on the basis of filling corresponding channels parameters and the table of network settings.

16 bits model systems have not a linked setup of communication channels with the user program. Therefore, during the central unit parameters setup it must be, after the **PLC** mode selection within the corresponding serial channel, set one of addresses presented in the network list (here 0, 1, 8 or 27). Indeed, all systems must have the same transmission rate set within the corresponding serial channel.

2.3.3. Network operation

Network operation runs on the background of the user program asynchronously with the cycle period of the user program. The actual network data are stored in the internal memory of the system and their interchange with the scratchpad takes place always during the I/O scan of the user program.

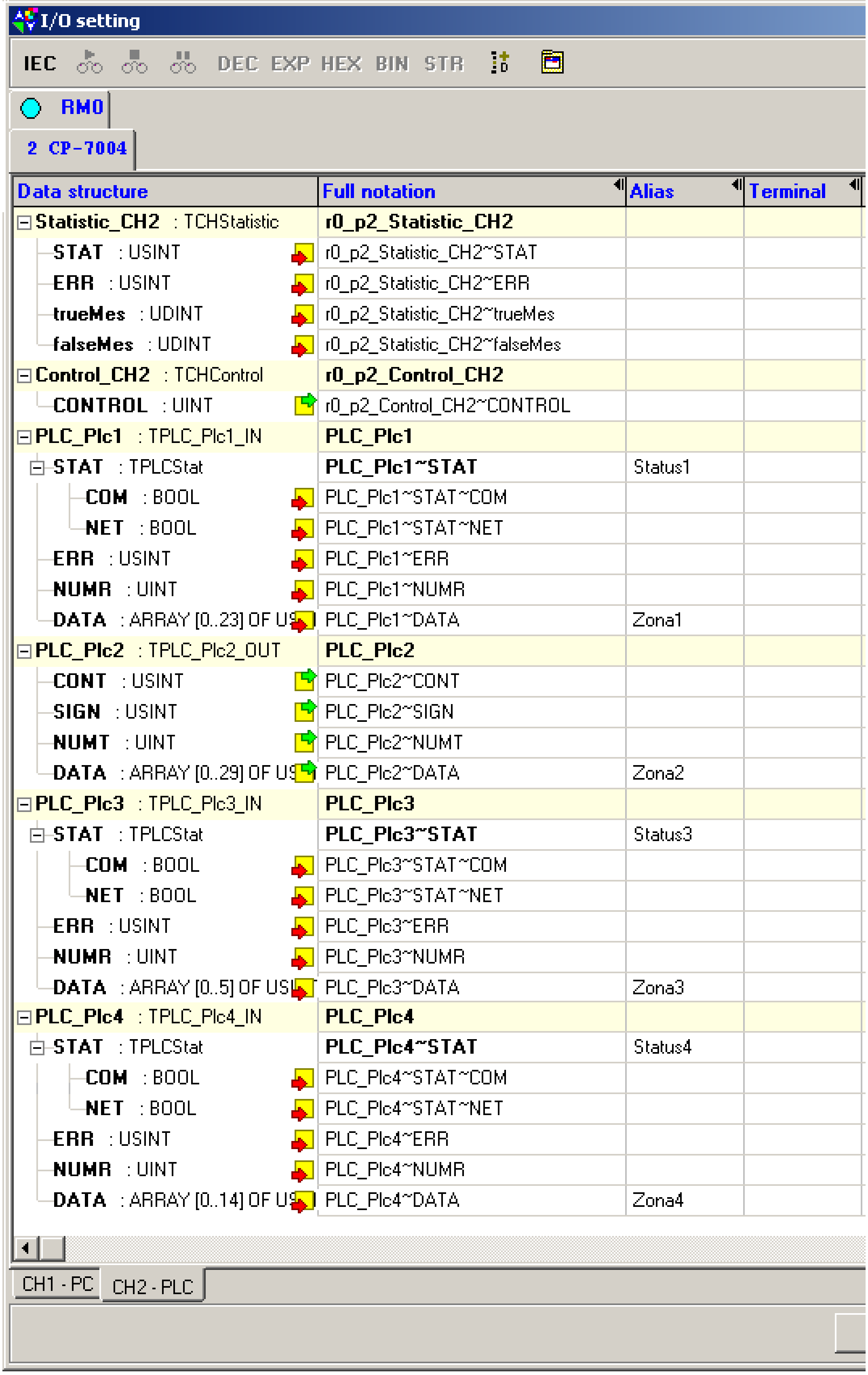
Operation diagnostics and network data

Serial channels in the **PLC** mode release the network status diagnostics data, data shared with individual participants and communication status of these participants. The data are stored in the scratchpad and are easily accessible in the *I/O setting* panel, accessible through icon IO (fig.2.3.7).

Data have assigned symbolic names which begin with the rack and position number. In the column *Full notation*, a concrete symbolic name for the given item is always specified.

If we want to use data in the user program, we use either this symbolic name or we enter our own symbolic name in the column *Alias* which we can then use. In no case do we use absolute operands as they can alter after a new compilation of the user program.

The user enters data intended for sharing to the relevant system passing zone for which the user program is designed. In other passing zones there are data transferred from other network users. It is necessary to check the COM bit in the relevant status before processing.



Data structure	Full notation	Alias	Terminal
⊟ Statistic_CH2 : TCHStatistic	r0_p2_Statistic_CH2		
STAT : USINT	r0_p2_Statistic_CH2~STAT		
ERR : USINT	r0_p2_Statistic_CH2~ERR		
trueMes : UDINT	r0_p2_Statistic_CH2~trueMes		
falseMes : UDINT	r0_p2_Statistic_CH2~falseMes		
⊟ Control_CH2 : TCHControl	r0_p2_Control_CH2		
CONTROL : UINT	r0_p2_Control_CH2~CONTROL		
⊟ PLC_Plc1 : TPLC_Plc1_IN	PLC_Plc1		
⊟ STAT : TPLCStat	PLC_Plc1~STAT	Status1	
COM : BOOL	PLC_Plc1~STAT~COM		
NET : BOOL	PLC_Plc1~STAT~NET		
ERR : USINT	PLC_Plc1~ERR		
NUMR : UINT	PLC_Plc1~NUMR		
DATA : ARRAY [0..23] OF US	PLC_Plc1~DATA	Zona1	
⊟ PLC_Plc2 : TPLC_Plc2_OUT	PLC_Plc2		
CONT : USINT	PLC_Plc2~CONT		
SIGN : USINT	PLC_Plc2~SIGN		
NUMT : UINT	PLC_Plc2~NUMT		
DATA : ARRAY [0..29] OF US	PLC_Plc2~DATA	Zona2	
⊟ PLC_Plc3 : TPLC_Plc3_IN	PLC_Plc3		
⊟ STAT : TPLCStat	PLC_Plc3~STAT	Status3	
COM : BOOL	PLC_Plc3~STAT~COM		
NET : BOOL	PLC_Plc3~STAT~NET		
ERR : USINT	PLC_Plc3~ERR		
NUMR : UINT	PLC_Plc3~NUMR		
DATA : ARRAY [0..5] OF USI	PLC_Plc3~DATA	Zona3	
⊟ PLC_Plc4 : TPLC_Plc4_IN	PLC_Plc4		
⊟ STAT : TPLCStat	PLC_Plc4~STAT	Status4	
COM : BOOL	PLC_Plc4~STAT~COM		
NET : BOOL	PLC_Plc4~STAT~NET		
ERR : USINT	PLC_Plc4~ERR		
NUMR : UINT	PLC_Plc4~NUMR		
DATA : ARRAY [0..14] OF US	PLC_Plc4~DATA	Zona4	

Fig.2.3.7 Serial channel data in ***PLC*** *mode*

Network diagnostics:
Input data *rx_py_Statistic_XXX* (*TCHStatistic* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Stat - network status (sets the **PLC** mode driver) (usint type)
0 - network start
1 - network operation monitoring
2 - establishing of connection with another network participants
3 - sending own data to the network
4 - passing network control to another participant
5 - network control take-over
6 - data reception
9 - suspension of communication (stand-by)
14 ($0E) - operation via Ethernet
Err - communication error (see chapter 3.6) (usint type)
trueMes - number of valid network communications (udint type)
falseMes - number of invalid network communications (udint type)

Output data *rx_py_Control_XXX* (*TCHControl* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Control - communication control (not used for now) (uint type)

Data provided for the network:
Output data *PLC_PPPNNN* (*TPLC_ PPPNNN_OUT* structure):

(PPP - Network code prefix (if any), NNN - project name)

Cont - communication control (not used for now) (usint type)
Sign - reserve (usint type)
NumT - length of data passed to the network (uint type)
Data [x] - data passed to the network (array element usint type)

Read data from the network:
Input data *PLC_PPPNNN* (*TPLC_PPPNNN_IN* structure):

(PPP - Network code prefix (if any), NNN - project name)

Stat - status of communication with the participant (8-times bool type)

	NET	X	X	X	X	X	COM	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

NET - participant
0 - a non-connectable station (station serial channel is set to another mode)
1 - network system with EPSNET-F protocol
COM - communication status
0 - communication not established, following data are not valid
1 - communication established, following data are valid
Err - error of communication with participant (see chapter 3.6) (usint type)
NumR - length of read data from the participant (uint type)
Data [x] - data read from the participant (array element usint type)

2.4. MPC MODE - DATA INTERCHANGE AMONG SLAVE AND MASTER SYSTEMS

The serial channel in this mode is used for connection of slave systems and several TECOMAT / TECOREG master systems for the purpose of data interchange among them. Communication is proceeding via the EPSNET protocol on the RS-485 serial interface that enables network creation.

Slave systems

The slave system can be any device with implemented EPSNET network services, thus, absolute majority of TECOMAT and TECOREG systems. The relevant serial channel of the slave system must be set to the **PC** mode.

Master systems

The initialisation of network passing zones is part of the user program of the master system only. The relevant serial channel of the slave system must be set to the **MPC** mode.

The **MPC** mode allows operation of more master systems within one network. Communication is performed on the query - response basis. The system in the **PC** mode behaves as a passive slave participant (slave), the system in the **MPC** mode behaves as an active master participant (master).

Data interchange is cyclic which means that the transmitted data are permanently refreshed by repeated communication. Moreover, the master participant allows based on a command from the user program sending of unrepeated communication and on the other hand, stopping data interchange with the network participant. This results in a significant expansion of possibilities of use of this network. The EPSNET network and its public functions are described in chapter 3.

Operation of more master systems

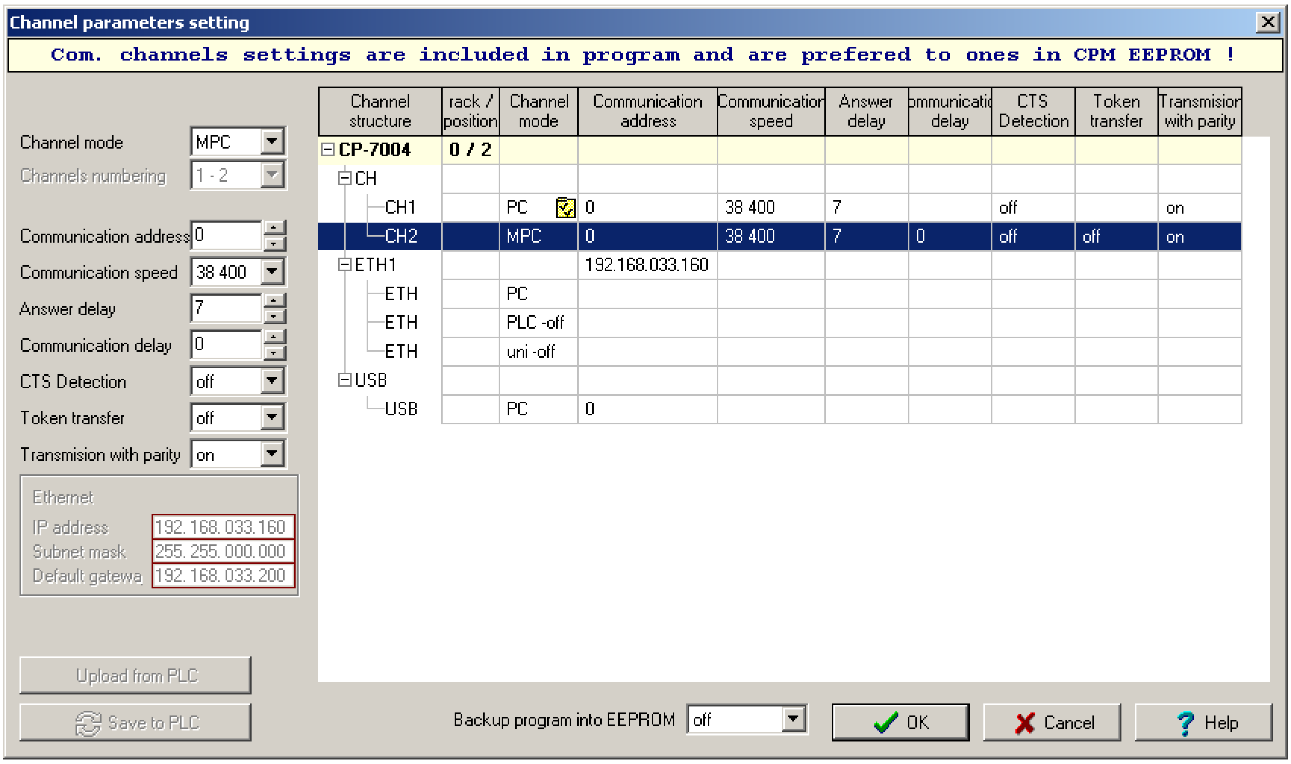
Each master system has a list in the user program with slave stations containing the request for reading and writing. Each slave station can be present on the lists of more master systems. A master system can also be present on the list of slave systems of another master system. The operation of more master systems within one network is realized by means of token telegram interchange. The station that receives token (all stations set to the **MPC** mode) behaves from that moment as a master and treats its requirements according to the list in the user program. Then it passes the token to another master station and from that moment it acts as a slave capable of fulfilling requirements of any other station that is acting as a master at that moment. Stations set to the **PC** mode cannot receive the token and so they are always slaves. The token passing principle allows insertion of a new master station to the network without any intervention to the user program of existing network participants (providing their requirements on data interchange do not change).

2.4.1. Channel parameters setup

We select in the Mosaic development environment folder *Hw* | *HW Configuration*. In the list of modules, we select the central unit or the communication module by the mouse, the channel of which we want to set. Then we press the push-button *Settings* or icon ☑ on the line on the right from the number position. The *Channel parameters setting* panel (fig. 2.4.1) is displayed. We set the corresponding serial channel to **MPC** mode and then we set transmission rate, address, response timeout, transmission delay, CTS signal detection, token passing mode and parity.

If we press the push-button *Save to PLC*, the settings will be stored in the central unit (does not apply for serial channels of communication modules). If we undertake a compilation of the user program, the communication settings will become a part of the user program and will be activated in the central unit at the restart of the user program.

Moreover, for communication modules, the option *Channels numbering* must be set and as a consequence the logic identification to individual channels is assigned (e.g. CH3, CH4). The central unit has the CH1 and CH2 channels fixly assigned.



Channel parameters setting

Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

Channel mode: MPC
Channels numbering: 1 - 2
Communication address: 0
Communication speed: 38 400
Answer delay: 7
Communication delay: 0
CTS Detection: off
Token transfer: off
Transmision with parity: on

Ethernet
IP address: 192. 168. 033. 160
Subnet mask: 255. 255. 000. 000
Default gatewa: 192. 168. 033. 200

Channel structure	rack / position	Channel mode	Communication address	Communicatior speed	Answer delay	ommunicatio delay	CTS Detection	Token transfer	Transmisior with parity
⊟ **CP-7004**	**0 / 2**								
⊟ CH									
CH1		PC ☑	0	38 400	7		off		on
CH2		MPC	0	38 400	7	0	off	off	on
⊟ ETH1			192.168.033.160						
ETH		PC							
ETH		PLC -off							
ETH		uni -off							
⊟ USB									
USB		PC	0						

Backup program into EEPROM: off

Fig.2.4.1 ***MPC*** *mode setting*

Address selection

Each network participant must have his own exclusive address. The addresses do not need to concur.

Transmission rate selection

The data interchange with a slave system is in progress during the I/O scan only. The cycle period of the slave system must be shorter than 500 ms, otherwise, the communication will be lost. Data in the collection zones located in the scratchpad of the master system are always updated during the I/O scan.

The **MPC** mode enables the transmission rate of up to 115.2 kBd. The higher the transmission rate, the shorter data transmission time, however, the line's resistance against disturbances decreases. We do recommend choosing only as high transmission rate that ensures and manages the data transfer in such time that we need considering the reaction requirements of the technology being controlled. By decreasing the transmission rate, the resistance against interference increases and, for some TECOMAT and

TECOREG systems, the cycle period shortens too. For slave systems, all conditions of the **PC** mode are valid (see chapter 2.2).

In CP-700x TC700 and CP-10xx FOXTROT central units the communication on the first two channels CH1 and CH2 and within central units is performed by the processor of the central unit.

Since the communication runs independently of the user program, it affects all cycle phases of the user program evenly. It applies here that by increasing the transmission rate the cycle period becomes longer. In table 2.4.1 the cycle period extension in per cents is shown.

Table.2.4.1 The average cycle period extension of the user program depending on the serial channel transmission rate

Transmission rate	TC650, CP-7001, CP-7002, CP-7003, CP-7005 TC700*	FOXTROT, CP-7004 TC700*
9.6 kBd	0.3 %	0.1 %
19.2 kBd	0.5 %	0.1 %
38.4 kBd	1.0 %	0.2 %
57.6 kBd	1.5 %	0.3 %
115.2 kBd	3.0 %	0.6 %

* Serial channels of communication modules SC-7101, SC-7102 extend the cycle period by transmission of usable data only. The communication itself is controlled by the local processor of the communication module.

Application of communication modules SC-7101, SC-7102

If we use, in the PLC TC700 for the central unit, the SC-7101 or SC-7102 communication module with other two communication channels, it is advisable to use one of these channels for the connection to the network. The communication load is then moved to the processor of the communication module and it stops influencing the cycle period.

Response timeout setting

Response timeout setup enables a selection of the minimum time period that elapses from sending the last message byte to the initiation of the first response byte transmission of the questioned system. During this period the master system must prepare for data acceptance. Sometimes this preparation can take more time depending on the type of the master system (e.g. on the operation system used on the PC). Usage of some serial interface converters, repeaters, modems and radio modems also requires a longer response timeout. Usually it is a device switching the direction of the communication (e.g. on the RS-485 interface) or causing a long transmission delay.

A minimum value of response timeout is set. Its maximum value, for most of function, depends on the cycle time of the system because data are passed on only during the I/O scan to ensure data constancy during user program execution. The minimum timeout response can be fixed from 1 to 99 ms or it can be set to 0 value which means that the minimum response timeout corresponds to the time required for sending one byte, thus, it is dependant on the transmission rate according to the Table 2.2.2.

Transmission delay setting

Optional transmission delay serves the resolution of cases when the master system is interconnected with slave systems via modems that are causing a communication delay, which when summed up with the maximum cycle period of any slave PLC exceeds

500 ms. Equally the transmission delay setup can be used in such cases where the cycle period of any of the slave PLC is longer than 500 ms.

The transmission delay is set using multiples of 100 ms and can take the value from 0 to 6.0 s. The 0 value signify that the master system anticipates the response within maximum of 0.5 s (the maximum cycle period of the slave system). Values 1 to 60 determine transmission delay from 0.1 s to 6.0 s which is added to the value 0.5 s mentioned already. Values 61 to 99 set the maximum transmission delay to 6.0 s.

Modems with automatic signal polarity setup

Some modems contains circuits for automatic setup of the transmitted signal polarity. To be set, these modems require idle condition on the line after operation start for about 1 to 2 s. Since the status must be handled right when the modem connects to the master system just after its operation initiation on the serial channel (disconnected cable to the modem, modem power supply failure, etc.), it is necessary the transmission delay of at least 2.0 s is set. After sending a message, the master system anticipate the response for 2.5 s, thus enabling automatic setting of an idle polarity of modem circuits. The following request of the master system will already be sent by the modem correctly.

CTS signal detection setting

Switching on the CTS signal detection enables to retain the external signal response from a modem. The response will be sent only after 10 ms following the CTS signal change to the corresponding RTS signal status valid for response sending.

CTS signal detection is primarily intended for the communication via modems.

The required solution to the advanced RTS setup before data broadcasting

A timeout of 10 ms after CTS change detection is intended to "calm" conditions within the transmission medium before data transmission starts (e.g. carrier frequency start). If the modem does not return the CTS signal but requires the 10 ms timeout between RTS signal setup and the data itself, we interconnect the RTS and CTS signals on the PLC channel connector and switch the CTS signal detection on.

Token passing mode

If a non-zero transmission delay is set (e.g. during the operation via modems) and there is only one master system on the network, this station, then by switching the token mode off we reduce management and accelerate data interchange. If there are more master stations on the network, token passing must not be switched off under any circumstances.

Parity setting

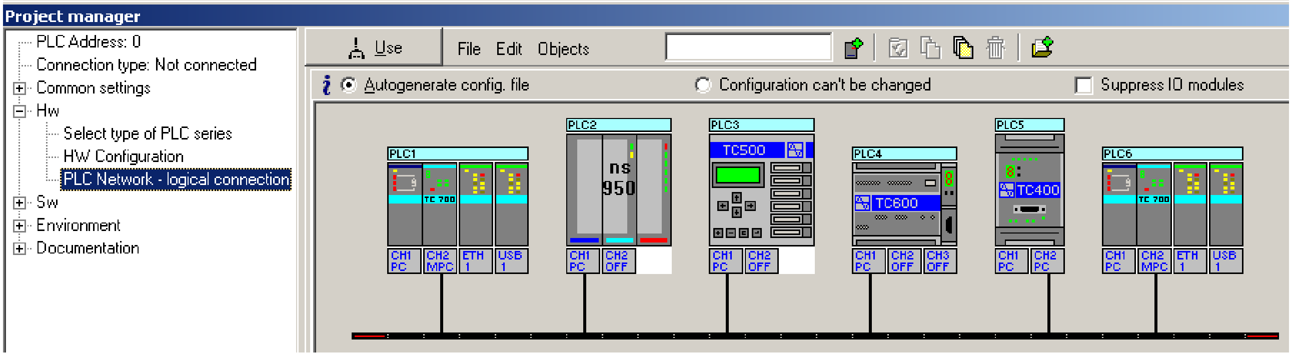
The EPSNET network protocol uses the even parity. However, some modems do not allow transmitting the parity. When used, the parity can be switched off on the serial channel of the central unit.

Attention! Parity considerably participates on the data transmission security. By its switching-off, the risk of undetected fault data transmission is increased. Possibilities of additional data protection are introduced in chapter 3.

2.4.2. Network initialisation

Parameters setup in the Mosaic environment

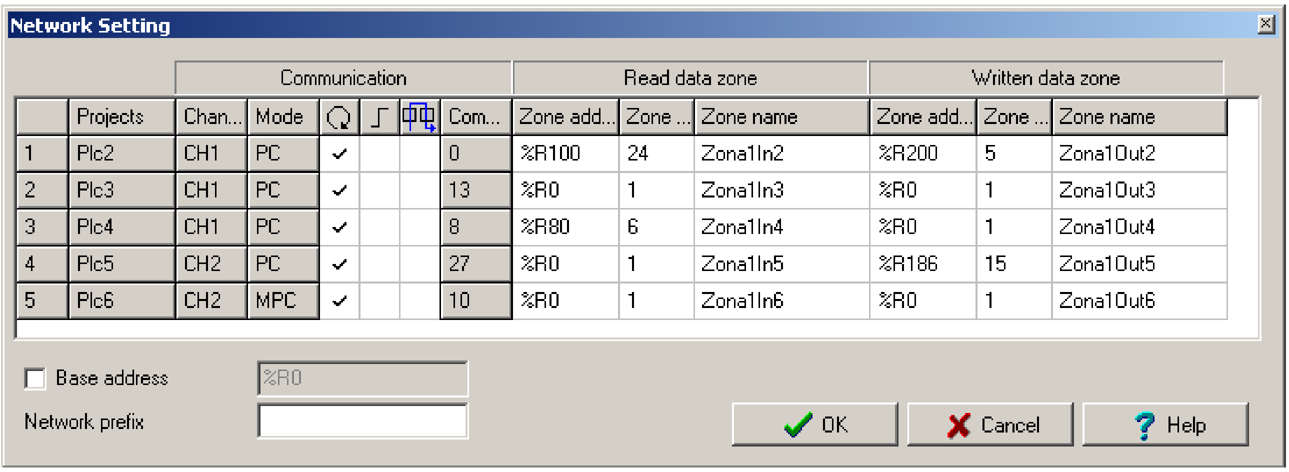
The assignment of serial channel zones into the scratchpad in the Mosaic development environment is accomplished by filling in the form on basis of which the initialization of the channel is set automatically. This form is accessible from the project manager folder *Hw* | *PLC Network* (fig.2.4.2). On the screen we insert all PLCs from the project group, draw a bus and interconnect to it all channels that we intend to have on one network. Using the right mouse click on the selected channel in the **MPC** mode in this network, a local menu appears where we select the item *Network setting* (fig. 2.4.3).


Project manager

- PLC Address: 0
- Connection type: Not connected
- Common settings
- Hw
  - Select type of PLC series
  - HW Configuration
  - PLC Network - logical connection
- Sw
- Environment
- Documentation

● Autogenerate config. file ○ Configuration can't be changed □ Suppress IO modules

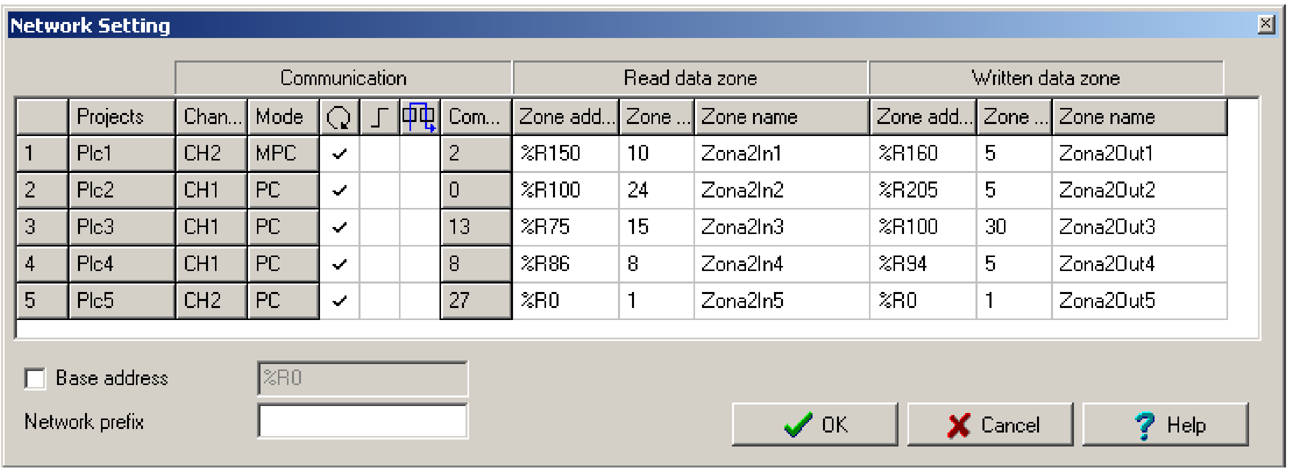
Fig.2.4.2 PLC network creation according to an example


Network Setting

		Communication						Read data zone			Written data zone		
	Projects	Chan...	Mode	⟳	∫	⇆	Com...	Zone add...	Zone ...	Zone name	Zone add...	Zone ...	Zone name
1	Plc2	CH1	PC	✓			0	%R100	24	Zona1In2	%R200	5	Zona1Out2
2	Plc3	CH1	PC	✓			13	%R0	1	Zona1In3	%R0	1	Zona1Out3
3	Plc4	CH1	PC	✓			8	%R80	6	Zona1In4	%R0	1	Zona1Out4
4	Plc5	CH2	PC	✓			27	%R0	1	Zona1In5	%R186	15	Zona1Out5
5	Plc6	CH2	MPC	✓			10	%R0	1	Zona1In6	%R0	1	Zona1Out6

□ Base address %R0

Network prefix


Network Setting

		Communication						Read data zone			Written data zone		
	Projects	Chan...	Mode	⟳	∫	⇆	Com...	Zone add...	Zone ...	Zone name	Zone add...	Zone ...	Zone name
1	Plc1	CH2	MPC	✓			2	%R150	10	Zona2In1	%R160	5	Zona2Out1
2	Plc2	CH1	PC	✓			0	%R100	24	Zona2In2	%R205	5	Zona2Out2
3	Plc3	CH1	PC	✓			13	%R75	15	Zona2In3	%R100	30	Zona2Out3
4	Plc4	CH1	PC	✓			8	%R86	8	Zona2In4	%R94	5	Zona2Out4
5	Plc5	CH2	PC	✓			27	%R0	1	Zona2In5	%R0	1	Zona2Out5

□ Base address %R0

Network prefix

Fig.2.4.3 Network parameters setting in Plc1 (addr. 2) (above) and in Plc6 (addr. 10) (below) according to an example

The collection network enables both the data reading from slave systems and data writing into these slave systems. If we only want to read data from the slave system, we set in the table the number of written data to 0. Analogically, if we only want to write data to the slave system, we set the number of read data to 0.

By ticking it is possible to select parameters marked by following icons:

↻ cyclic communication - data are transferred constantly

⎍ one-time communication - data are transferred just once according to user program request

⇆ message transfer from other networks enabled - the selection activates the function of message transfer among communication channels assigned to this station which address must not collide with any other address within all other networks (chapter 4.)

If we tick *Base address of zones* item, we can set in the following field a physical address of data location in the master system. Otherwise, the physical location of data will be determined by the compiler.

In the field *Network prefix* we can enter network name that will be then added as the prefix to all symbolic names of the network. Its use is of importance in case when we realize more networks among same systems, thus, we can differentiate them from each other.

If we do a right-click on the line of the table, the local offer that allows following functions will appear (fig. 2.4.4):

Duplicate - add another line with the address corresponding to the address of the slave station that allows to define another read and written data area of the station

Delete - remove the selected line

Broadcast

- *Append* - add the line with a global address that allows to define data area written to all stations within the network
- *Delete* - remove the line with the global address

More detailed information is stated in following paragraphs.

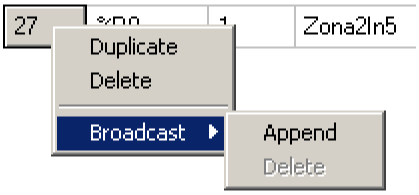


Fig.2.4.4 Local offer of the network parameters table

Table.4.2 Maximum number of transmitted data (read + entered) among the master system and one slave system depending on the total number of operated slave systems

Total number of operated slave systems	Max. number of transferred data with one slave system TC650, CP-700x TC700, FOXTROT
1 up to 64	238 + 238 bytes

Note: Exceeding the number of data results in error 83 cc 4206 (cc stands for the number of serial channels used)

Principles of addressing

The addresses of all systems in the network can have any value within the range of 0 to 99. These addresses do not need to follow-up each other and they must not coincide. Naturally, all systems must have the same transmission rate set on the corresponding serial channel

Collection of more areas from one system

If we need to transmit data from one slave system from two discontinuous areas (e.g. from registers R0 to R15 and registers R324 to 355), we list this slave system in the initialisation table twice, once with the first area and once with the second area. Then, the master system considers the slave system as two slave systems. However, considering the time convenience, it is more advantageous to merge both areas to one continuous.

Equally, we proceed in case we need to transmit more data then one communication allows.

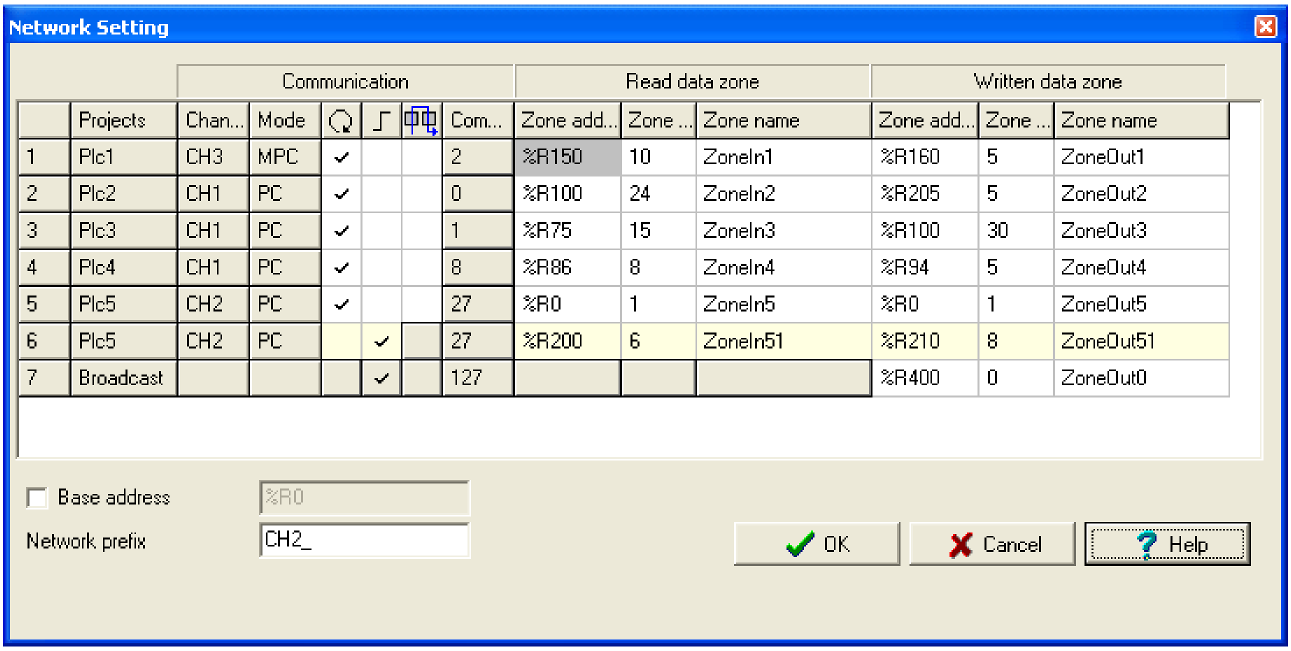
In the Mosaic development environment we create a next line defining communication with the same participant in such a way that on the line defining first communication we press the right mouse push-button and select *Duplicate* from the menu (fig.2.4.4). A new line is inserted, defining the same participant. The line can be edited as required (fig.2.4.5).

Data entry into all systems in the network (broadcast)

If we need to transfer same data from the master system to all other system within the network, we use so called Broadcast communication.

In the Mosaic environment we create the line defining broadcast communication using the right-mouse click on any line and select from the offer the command *Broadcast | Append* (fig.2.4.4). The new line is inserted, defining with the global address 127, that enables to define data broadcasted (fig.2.4.5).

Broadcast communication is always one-stoked. It serves mostly to fast data transfer to the whole network. It is necessary to realize that the address of the data zone set in this communication **is valid for all stations physically connected to the network!**



Network Setting

		Communication						Read data zone			Written data zone		
	Projects	Chan...	Mode	⟳	⎍		Com...	Zone add...	Zone ...	Zone name	Zone add...	Zone ...	Zone name
1	Plc1	CH3	MPC	✓			2	%R150	10	ZoneIn1	%R160	5	ZoneOut1
2	Plc2	CH1	PC	✓			0	%R100	24	ZoneIn2	%R205	5	ZoneOut2
3	Plc3	CH1	PC	✓			1	%R75	15	ZoneIn3	%R100	30	ZoneOut3
4	Plc4	CH1	PC	✓			8	%R86	8	ZoneIn4	%R94	5	ZoneOut4
5	Plc5	CH2	PC	✓			27	%R0	1	ZoneIn5	%R0	1	ZoneOut5
6	Plc5	CH2	PC		✓		27	%R200	6	ZoneIn51	%R210	8	ZoneOut51
7	Broadcast				✓		127				%R400	0	ZoneOut0

□ Base address %R0

Network prefix CH2_

Fig.2.4.5 Example of duplicated line and broadcast communication

Example

Let us realize two data collections in one network according to the following requirements:

Master system TC700 addr.2
Slave systems:
1. NS950 addr. 0, reading 24 bytes from register R100, writing 5 bytes from register R200
2. TC600 addr. 8, reading of 6 bytes from register R80, no writing
3. TC400 addr. 27, no reading, writing of 15 bytes from register R186

Master system TC700 addr.10
Slave systems:
1. NS950 addr. 0, reading 24 bytes from register R100, writing 5 bytes from register R205
2. TC500 addr. 1, reading 15 bytes from register R75, writing 30 bytes from register R100
4. TC700 addr. 2, reading 10 bytes from register R150, writing 5 bytes from register R160
4. TC600 addr. 8, reading of 8 bytes from register R86, writing 5 bytes from register R94

In Mosaic, the tables will be created automatically according to fig. 2.4.2 and 2.4.3.

The structure of the communication zone in the scratchpad of master systems will be as follows:

	addr. 2
R0 R149	
R150 R164	data exchanged with system addr. 10 - see next column
R165 ...	
ZoneIn2	values read from registers R100 to R123 of system addr. 0
ZoneOut2	values written to registers R200 to R204 of system addr. 0
ZoneIn4	values read from registers R80 to R85 of system addr. 8
ZoneOut5	values written to registers R186 to R200 of system addr. 27

	addr. 10
R0 ...	
ZoneIn1	values read from registers R100 to R123 of system addr. 0
ZoneOut1	values written to registers R205 to R209 of system addr. 0
ZoneIn2	values read from registers R75 to R89 of system addr. 1
ZoneOut2	values written to registers R100 to R129 of system addr. 1
ZoneIn3	values read from registers R150 to R159 of system addr. 2
ZoneOut3	values written to registers R160 to R164 of system addr. 2
ZoneIn4	values written to registers R86 to R93 of system addr. 8
ZoneOut4	values written to registers R94 to R98 of system addr. 8

It results from the aforesaid that the contents of individual zones of the slave systems are stored immediately one after each another to the communication zone of the corresponding master system in the order-read data and written data. It is vital to keep in mind that the communication zone should not coincide with any other structures. This can be easily avoided by symbolic declarations of registers.

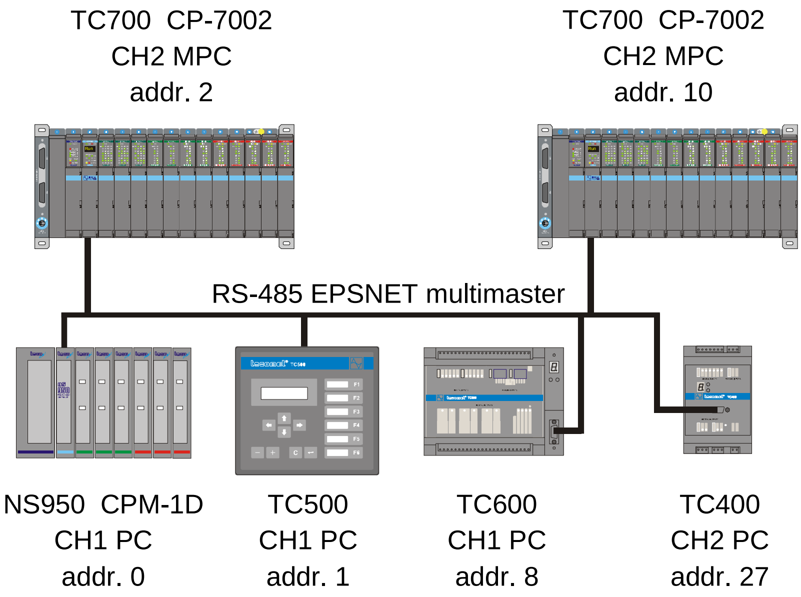


Fig.2.4.6 Interconnection of network systems according to the example (master systems have the serial channels set in the ***MPC*** *mode, slave systems have the serial channel set in the* ***PC*** *mode)*

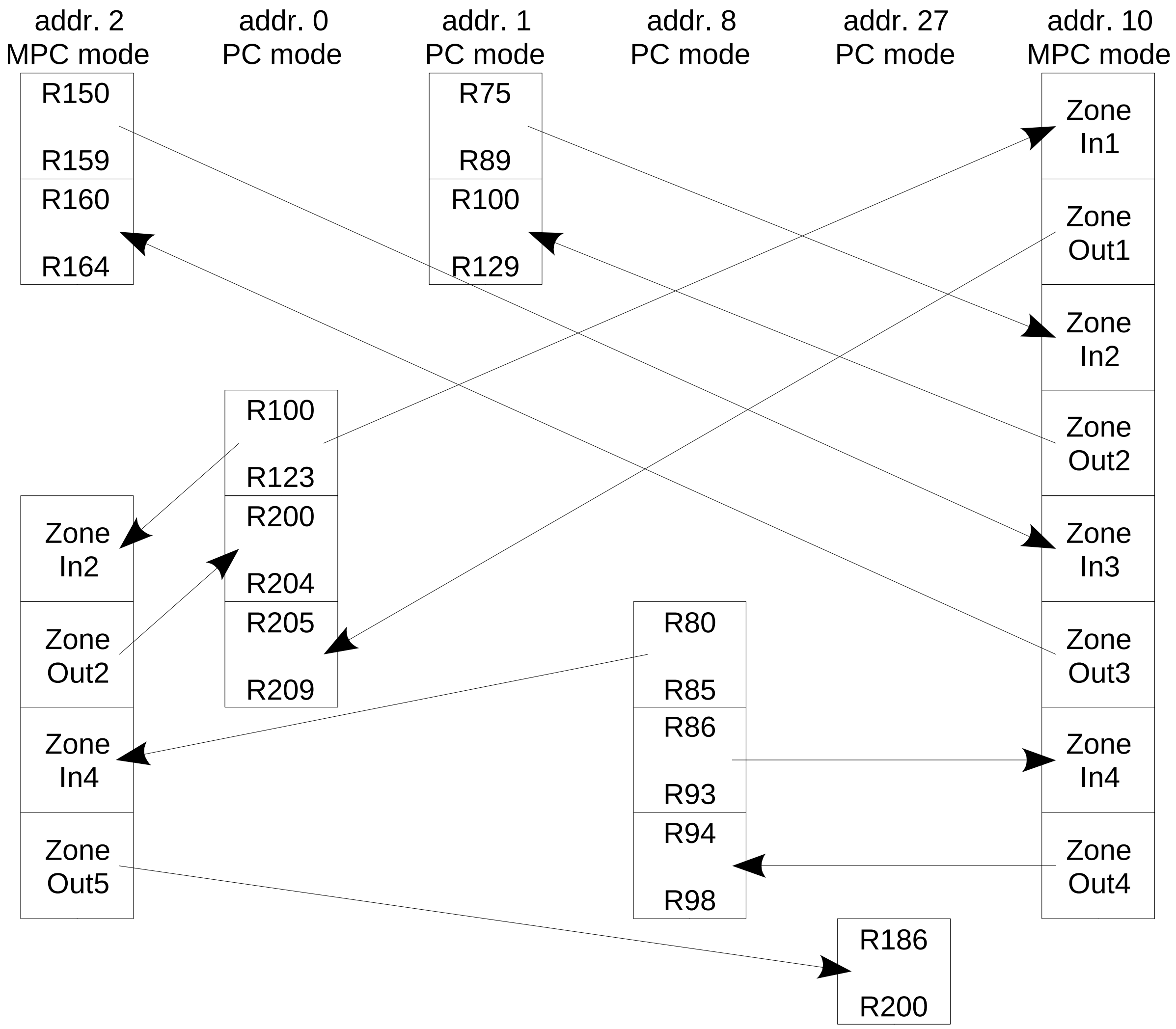


Fig.2.4.7 Graphic representation of data interchange among master systems in the ***MPC*** *mode and slave systems in the* ***PC*** *mode (the absolute address of data zones is given by compilation)*

2.4.3. Network operation

Network operation runs on the background of the user program asynchronously with the cycle time of the user program. The current network data are stored in the internal memory of the system and their interchange with the scratchpad proceeds always at the I/O scan of the user program.

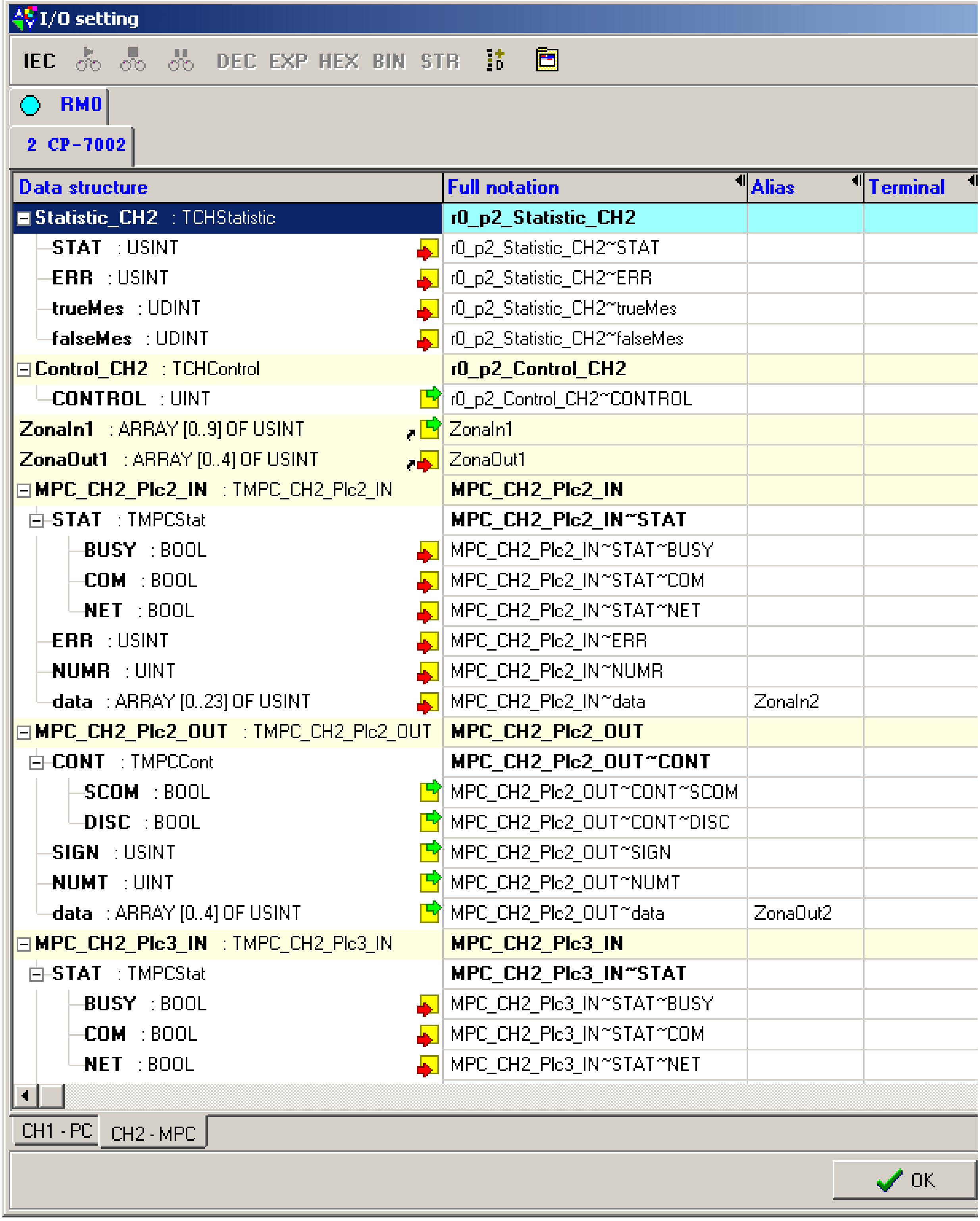
Operation diagnostics and network data

The serial channels in the **MPC** mode release diagnostics data of the network status, data exchanged with individual participants and communication status among these participants. These data are stored in the scratchpad and are easily accessible via panel *I/O Setting*, accessible through icon IO (fig.2.4.8).

Data have assigned symbolic names which begin with the rack and position number. In the column *Full notation*, a concrete symbolic name for the given item is always specified.

If we want to use these data in the user program, we use either this symbolic name or we enter our own symbolic name in the column *Alias* which we then can use. In no case do we use absolute operands since they alter after a new compilation of the user program.

Into the zones with output data the user writes data to be written into corresponding slave stations. Before data processing it is necessary to check the COM bit at the relevant status.



Data structure	Full notation	Alias	Terminal
Statistic_CH2 : TCHStatistic	**r0_p2_Statistic_CH2**		
STAT : USINT	r0_p2_Statistic_CH2~STAT		
ERR : USINT	r0_p2_Statistic_CH2~ERR		
trueMes : UDINT	r0_p2_Statistic_CH2~trueMes		
falseMes : UDINT	r0_p2_Statistic_CH2~falseMes		
Control_CH2 : TCHControl	**r0_p2_Control_CH2**		
CONTROL : UINT	r0_p2_Control_CH2~CONTROL		
ZonaIn1 : ARRAY [0..9] OF USINT	ZonaIn1		
ZonaOut1 : ARRAY [0..4] OF USINT	ZonaOut1		
MPC_CH2_Plc2_IN : TMPC_CH2_Plc2_IN	**MPC_CH2_Plc2_IN**		
STAT : TMPCStat	**MPC_CH2_Plc2_IN~STAT**		
BUSY : BOOL	MPC_CH2_Plc2_IN~STAT~BUSY		
COM : BOOL	MPC_CH2_Plc2_IN~STAT~COM		
NET : BOOL	MPC_CH2_Plc2_IN~STAT~NET		
ERR : USINT	MPC_CH2_Plc2_IN~ERR		
NUMR : UINT	MPC_CH2_Plc2_IN~NUMR		
data : ARRAY [0..23] OF USINT	MPC_CH2_Plc2_IN~data	ZonaIn2	
MPC_CH2_Plc2_OUT : TMPC_CH2_Plc2_OUT	**MPC_CH2_Plc2_OUT**		
CONT : TMPCCont	**MPC_CH2_Plc2_OUT~CONT**		
SCOM : BOOL	MPC_CH2_Plc2_OUT~CONT~SCOM		
DISC : BOOL	MPC_CH2_Plc2_OUT~CONT~DISC		
SIGN : USINT	MPC_CH2_Plc2_OUT~SIGN		
NUMT : UINT	MPC_CH2_Plc2_OUT~NUMT		
data : ARRAY [0..4] OF USINT	MPC_CH2_Plc2_OUT~data	ZonaOut2	
MPC_CH2_Plc3_IN : TMPC_CH2_Plc3_IN	**MPC_CH2_Plc3_IN**		
STAT : TMPCStat	**MPC_CH2_Plc3_IN~STAT**		
BUSY : BOOL	MPC_CH2_Plc3_IN~STAT~BUSY		
COM : BOOL	MPC_CH2_Plc3_IN~STAT~COM		
NET : BOOL	MPC_CH2_Plc3_IN~STAT~NET		

*Fig.2.4.8 Serial channel data in the **MPC** mode (according to the example for PLC addr. 2)*

Network diagnostics:
Input data *rx_py_Statistic_XXX* (*TCHStatistic* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Stat - network status (sets the **MPC** mode driver) (usint type)
0 - network initialization
1 - network acceptance and verification
2 - acceptance processing and waiting for reply
3 - connection establishment (logging in)
4 - data interchange
5 - life list verification
6 - submission of the network control to another participant
7 - undertaking the network control
8 - undertaking of the message from another network
9 - suspension of communication (stand-by status)

Err - communication error (see chapter 3.6) (usint type)

trueMes - number of valid network communications (udint type)

falseMes - number of invalid network communications (udint type)

Output data *rx_py_Control_XXX* (*TCHControl* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Control - communication control (not used for now) (uint type)

Data exchanged with the master participant:
Input data *ZoneIn1* (usint array):

user data itself received from another master station

Output data *ZoneOut1* (usint array):

user data itself sent to another master station

Both zones exist only when there are more master stations in the network and at least one of them is programmed for communication with this station. Names of these zones are then determined by symbolic naming in the table *Network setting* (fig. 2.4.3) relevant to the master station.

Data exchanged with slave participants:
Input data *MPC_XXX_NNN_IN* (*TMPC_ XXX_NNN_IN* structure):

(XXX - communication channel designation, NNN - project name)

Stat - communication status with the participant (8-times bool type)

	NET	X	X	X	X	X	COM	BUSY
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

NET - participant
0 - a non-connectable station (the station serial channel is set to another mode)
1 - slave system of connection network

COM - communication status
0 - communication not established, following data ate not valid
1 - communication established, following data are valid
BUSY- unrepeated communication status
0 - unrepeated communication does not proceed
1 - unrepeated communication proceeds, data not loaded yet

Err - communication error with participant (see chapter 3.6) (usint type)
NumR - length of read data from participant (uint type)
Data [x] - data read from participant (usint type array element)

Output data *MPC_XXX_NNN_OUT* (*TMPC_ XXX_NNN_OUT* structure):

(XXX - communication channel designation, NNN - project name)

Cont - communication control (8-times bool type)

	0	0	0	0	0	0	DISC	SCOM
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

DISC - communication suspension
0 - communication proceeds
1 - communication is suspended, data are not refreshed
SCOM - unrepeated communication execution
0 - unrepeated communication will not be proceeded
1 - unrepeated communication activation

Sign - reserve (usint type)
NumT - length of written data from participant (uint type)
Data [x] - data written from participant (usint type array element)

Cyclic data exchange

Data are permanently refreshed according to parameters specified in the initialisation. In the communication zone of the master system, there are data transmitted from the slave systems of the network. It is necessary to check the COM bit in the relevant status before their processing. If it is in log.1, data are valid.

If we want the data exchange to be ceased temporarily then we set the DISC bit within the control byte to log.1. While the data exchange is suspended, the master system does not proceed any communication with the corresponding slave system (the exception is token telegram pass-over or assumed message from another network). By setting the DISC bit to zero, the data exchange starts again.

Unrepeated data exchange

If the data exchange with the relevant slave system is initialised as unrepeated data exchange then this communication is started after setting the SCOM bit in the control byte to log.1. Immediately before inserting of this communication into the list, the SCOM bit is set to zero and the BUSY bit in the status is set to log.1. On the next occasion the unrepeated communication is inserted among cyclic communications already running. After its accomplishment the BUSY bit is set to zero and according to the result of the communication the bit COM is set. If the value is log.1, data are valid. At the corresponding part of the communication zone of the master system, data are unrepeatedly transmitted from the slave system.

In practise this means that we set the SCOM bit to log.1 and during the following cycle of the user program we test the BUSY bit. If the log. is 1, we repeat the test in the following cycle. If the BUSY bit is in log.0, we check the COM bit then. If the COM bit is in log.1, then the data interchange passed without any errors and possible received data are valid.

If we want to cease the possibility of the unrepeated data exchange temporarily, we set the DISC bit in the control byte to log.1. Meanwhile, the master system does not perform any communication with the slave system (the exception is token telegram pass-over or assumed message from another network). By setting the DISC bit to zero, the data exchange is enabled again.

2.5. UNI MODE - GENERAL USER CHANNEL

The application of the general user channel

The general user channel mode is intended for the universal use. It contains functions for the support of single-character oriented serial protocols, by which the channel can be set in such a way that it passes only valid data without the necessity of further user program checks.

The serial channel in this mode is freely usable for example for interconnection of serial printers, bar code scanners, frequency converters, intelligent sensors or operation panels.

The Ethernet interface in this mode enables the data interchange with the general device via UDP or TCP protocol.

Functions available for message acceptance via serial line

Functions available for message acceptance are as follows:

Detection of message initiation	- initial character test
Detection of message termination	- terminal character test
	- test during the idle condition on the line
	- fixed message length
	- getting the message length from message protocol
Station address detection	
Options for message protection	- parity check
	- test of message data checksum

Functions available for message posting via serial line

Functions available for message posting are as follows:

Options for message initiation	- initial character
Options for message termination	- terminal character
	- the idle condition on the line
Completion of the station address	
Options for message protection	- parity calculation
	- calculation of message data checksum

Individual functions and their options can be combined so that it is possible to approach the ideal state when the system receives for processing only data themselves without any further characters serving only for protection of message transmission.

The serial channel initialisation is a part of the user program. The relevant serial channel must be set to the **UNI** mode. The transmission rate and the address are part of the channel initialisation in the user program.

Selection of the serial channel transmission rate

Central units in this mode enable the transmission rate of up to 115.2 kBd. The higher the transmission rate, the shorter the data transmission. Since the communication is running independently of the user program, it affects all cycle phases of the user program evenly. It applies here that by increasing the transmission rate, the cycle time becomes slightly longer. In the table 2.5.1 the cycle time extension in per cent is listed. These values are valid for the maximum channel load (continuous communication). The longer timeouts between the messages, the lower the average cycle time extension will be.

Table 2.5.1 Average extension of the user program cycle time depending on the serial channel transmission rate and central unit type

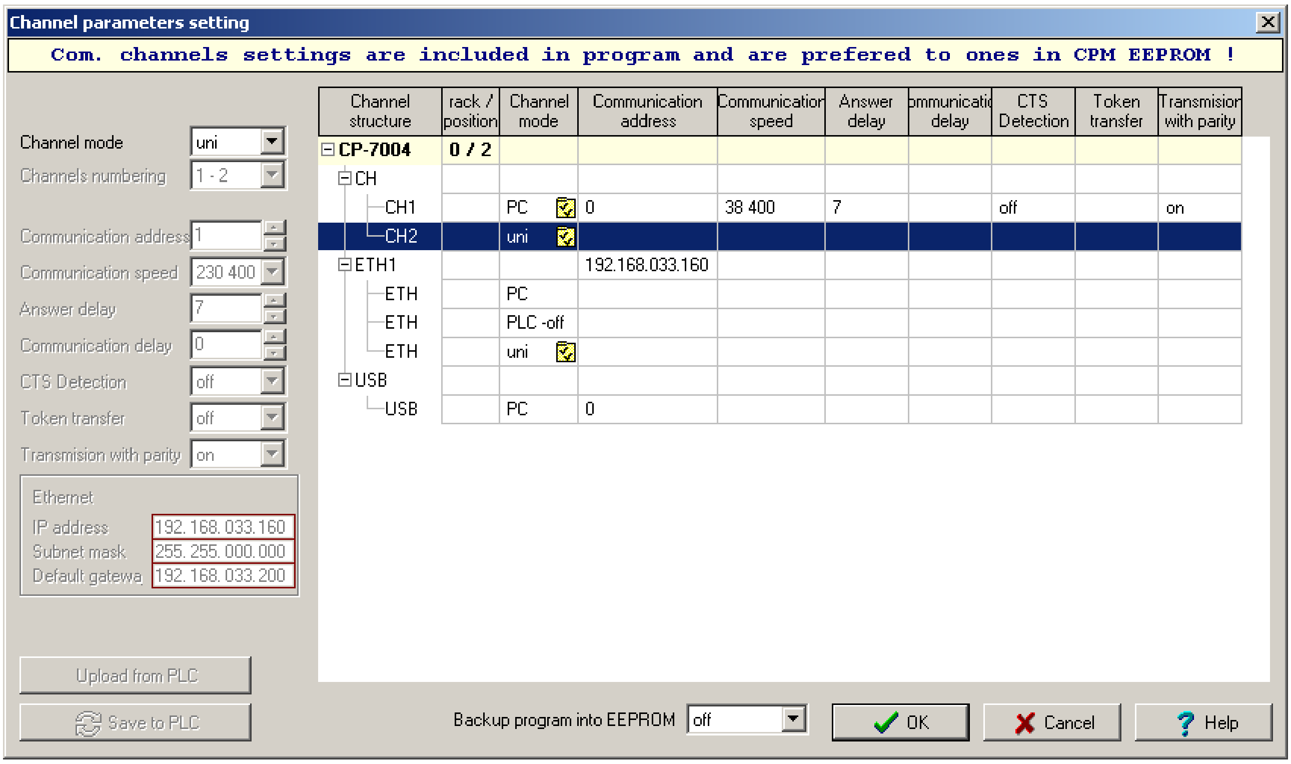
Transmission rate	TC650, CP-7001, CP-7002, CP-7003, CP-7005 TC700	FOXTROT, CP-7004 TC700
9.6 kBd	0.3 %	0.1 %
19.2 kBd	0.5 %	0.1 %
38.4 kBd	1.0 %	0.2 %
57.6 kBd	1.5 %	0.3 %
115.2 kBd	3.0 %	0.6 %

Ethernet interface configuration

Provided that the UNI mode is set within the Ethernet interface then the transmission of up to 1350 bytes via the general UDP or TCP protocol is enabled. In case of TCP protocol it is vital to asses whether the PLC should create the connection with the opposite station (master setup) or wait till the connection is established by the opposite station (slave setup). No function for message editing is in this case available.

2.5.1. Communication channel parameters setup

In the Mosaic project manager we select folder *Hw | HW Configuration*. In the list of individual modules we highlight, using the mouse, the central unit or the communication module, the channel of which we want to set. Then we press the push-button *Settings* or the icon ☑ on the line on the right from the position number. The *Channel parameters setting* panel will appear (fig. 2.5.1).



Channel parameters setting

Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

Channel mode: uni
Channels numbering: 1 - 2
Communication address: 1
Communication speed: 230 400
Answer delay: 7
Communication delay: 0
CTS Detection: off
Token transfer: off
Transmision with parity: on

Ethernet
IP address: 192. 168. 033. 160
Subnet mask: 255. 255. 000. 000
Default gatewa: 192. 168. 033. 200

Channel structure	rack / position	Channel mode	Communication address	Communication speed	Answer delay	ommunicati delay	CTS Detection	Token transfer	Transmision with parity
⊟ CP-7004	0 / 2								
⊟ CH									
CH1		PC ☑	0	38 400	7		off		on
CH2		uni ☑							
⊟ ETH1			192.168.033.160						
ETH		PC							
ETH		PLC -off							
ETH		uni ☑							
⊟ USB									
USB		PC	0						

Backup program into EEPROM: off

Fig.2.5.1 ***UNI*** *mode parameters setup*

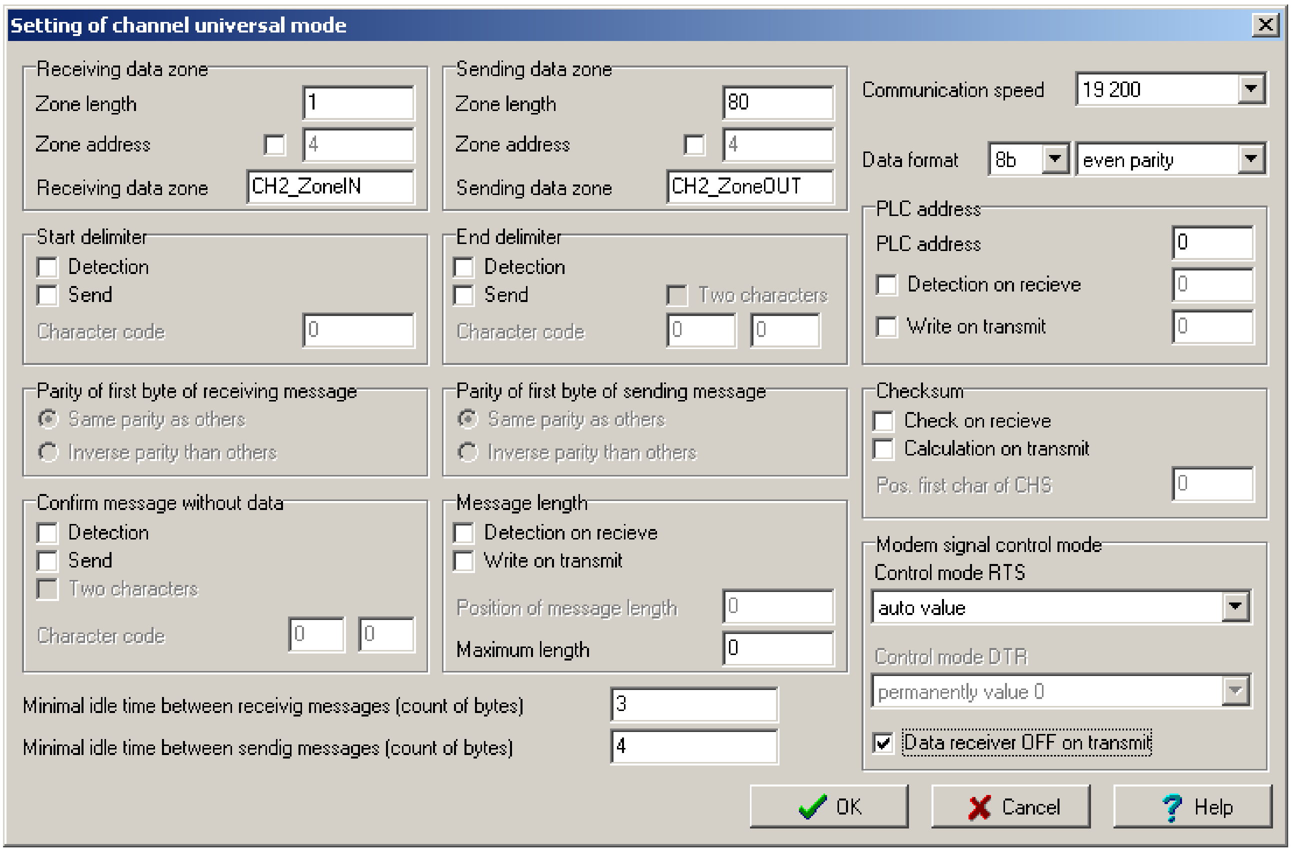
We set the corresponding serial channel to **UNI** mode. We must within communication modules to set the option *Channels numbering* and thus assign a logical marking to serial channels (e. g. CH3, CH4). The central unit has serial channels CH1 and CH2 fixly assigned.

We must activate the **UNI** mode (**UNI-off** is changed to **UNI**) within Ethernet interface. Further the IP address and IP mask must be saved to PLC.

Then we press ☑ button on the line of the selected communication channel and a window *Setting of channel universal mode* (fig.2.5.2) will appear. Set parameters for serial channels and for the Ethernet interface differ essentially and therefore are described separately in following chapters 2.5.1.1. and 2.5.1.2.

2.5.1.1. Serial channels

By pressing the ☑ icon on the line of the serial channel that is set to UNI mode, a window *Setting of channel universal mode* will appear. (fig.2.5.2).



Setting of channel universal mode

Receiving data zone
Zone length: 1
Zone address: □ 4
Receiving data zone: CH2_ZoneIN

Sending data zone
Zone length: 80
Zone address: □ 4
Sending data zone: CH2_ZoneOUT

Communication speed: 19 200
Data format: 8b, even parity

Start delimiter
□ Detection
□ Send
Character code: 0

End delimiter
□ Detection
□ Send
□ Two characters
Character code: 0 0

PLC address
PLC address: 0
□ Detection on recieve: 0
□ Write on transmit: 0

Parity of first byte of receiving message
● Same parity as others
○ Inverse parity than others

Parity of first byte of sending message
● Same parity as others
○ Inverse parity than others

Checksum
□ Check on recieve
□ Calculation on transmit
Pos. first char of CHS: 0

Confirm message without data
□ Detection
□ Send
□ Two characters
Character code: 0 0

Message length
□ Detection on recieve
□ Write on transmit
Position of message length: 0
Maximum length: 0

Modem signal control mode
Control mode RTS: auto value
Control mode DTR: permanently value 0
☑ Data receiver OFF on transmit

Minimal idle time between receivig messages (count of bytes): 3
Minimal idle time between sendig messages (count of bytes): 4

*Fig.2.5.2 **UNI** mode parameters settings for the serial channel*

Settings meaning and possibilities of individual items are as follows:

Receiving data zone

Zone length - Length of data section of a receiving zone of the channel.

Zone address - Index of an R register in which the receiving zone starts. If the check box before the entry is enabled, the entry can be changed, otherwise it is generated automatically. (automatically generated base address is not displayed in this field).

Receiving data zone - Symbolic name of the receiving zone.

Sending data zone

Zone length - Length of data section of a sending zone of the channel.
Zone address - Index of register R in which the sending zone starts. If the check box before the entry is enabled, the entry can be changed, otherwise it is generated automatically. (automatically generated base address is not displayed in this field).
Sending data zone - Symbolic name of the sending zone.

Communication speed

Selection of the transmission rate in Bd.

Data format

The number of transmitted bits within one data byte (8 or 7 bits).

Parity

Parity setup:
Without parity - parity bit is not transmitted (only for data format of 8 bits)
Odd parity - the value of the parity bit is at such value to ensure that the number of ones of data bits and the parity bit in one byte is odd
Even parity - the value of the parity bit is at such value to ensure that the number of ones of data bits and the parity bit in one byte is even.
Parity permanently 0 - the value of the parity bit is permanently log.0
Parity permanently 1 - the value of the parity bit is permanently log.1

Each message byte has 1 stop bit. If 2 stop bits are required and parity is not at the same time, the first stop bit can be simulated using an option parity log.1. The second stop bit is added automatically. If 2 stop bits and parity, too, are required, the realization is possible only for 7 bits format. We set the format of 8 bits, parity log.1. The parity must be calculated by the user program and entered to the highest bit of the byte (bit 7).

It is not possible to set the format of 7 bits without parity. The parity can be switched off only for the format of 8 bits.

Start delimiter

Detection - Activation of the detection option of receiving message initial character.
Send - Activation of the generation option of the initial character on the beginning of the message sent.
Character code -If at least one of options *Detection* or *Send* the initial character is activated, it is possible to set the value of this character.

End delimiter

Detection - Activation of the detection option of receiving message terminal character.
Send - Activation of the generation option of the terminal character on the beginning of the message sent.
Two characters - If the check box is enabled, the terminal character has two characters otherwise it has one character.
Character code - If at least one of options *Detection* or *Send* the terminal character is activated, it is possible to set the value of this character. If the check box *Two characters* is enabled, the terminal character has two characters and both of them can be set.

PLC address

PLC address - Station address on the serial channel (range 0 to 255).
Detection on receive - Activation of the address detection option for the message received. If this option is enabled, the address position in the received message can be set.
Write on transmit - Activation of the address generation option for the message sent. If this option is enabled, the address position in the sent message can be set.

Check sum

Check on receive - Activation of the check option of the received message sum check.
Calculation on transmit - Activation of the generation option of the sent message check sum.
Pos. first char. of CHS - If at least one of options *Check on receive* or *Calculation on transmit* is enabled, it is possible to set the position of the first character included in the check sum. The check sum is expected at or is added to the end of the message. If the terminal character is detected or generated, the check sum is expected at or more precisely generated immediately before this happens.

Confirm message without data

Detection - Activation of the detection option of a confirmation character
Send - Activation of the generation option of a confirmation character
Two characters - If the check box is enabled, the confirmation character has two characters otherwise it has one character
Character code - If at least one of options *Detection* or *Send* the confirmation character is activated, it is possible to set the value of this character. If the check box *Two characters* is enabled, the confirmation character has two characters and both characters can be set

Message length

Detection on receive - Activation of the length detection option of the received message
Write on transmit - Activation of the length detection option of the sent message
Position of message length - If at least one of options *Detection on receive* or *Write on transmit* is enabled, it is possible to set the position of message length data. Message length includes all characters following length data except the check sum and the terminal character
Maximum length - The value determining maximum possible length of the message received. Received characters exceeding this length will be considered as a new message. If the value of this parameter is 0, it is set internally to the value corresponding to the length of the receiving zone in the scratchpad

Modem signal control mode

Control mode RTS - RTS modem signal control mode setup
permanently value 0 - the signal has the permanent value log.0
permanently value 1 - the signal has the permanent value log.1
auto value - The signal is controlled by the serial channel transmitter, it has the value log.0 during transmission
depends on SIGN.1 (from program) - The signal is controlled from the PLC user program by the relevant bit

auto value with CTS detection - The signal is controlled by the serial channel transmitter, it has the value of log.0 during transmission, and the transmission is conditioned by CTS signal setting which must be log.0.

Data receiver off on transmit - If the check box is enabled, the receiver is switched off during sending. If any device causing echoing of sent characters is connected, their unwanted reception will be avoided.

Minimal idle time between received messages (count of bytes)

The minimum idle period on the line corresponding to the number of received bytes which is defined between two received messages. After the expiration of this time, the message received is considered to be complete. This parameter allows the reception of messages of various lengths. If this parameter is 0, the characters received are not comprised into integrated messages but they are passed to the PLC scratchpad immediately at each program cycle.

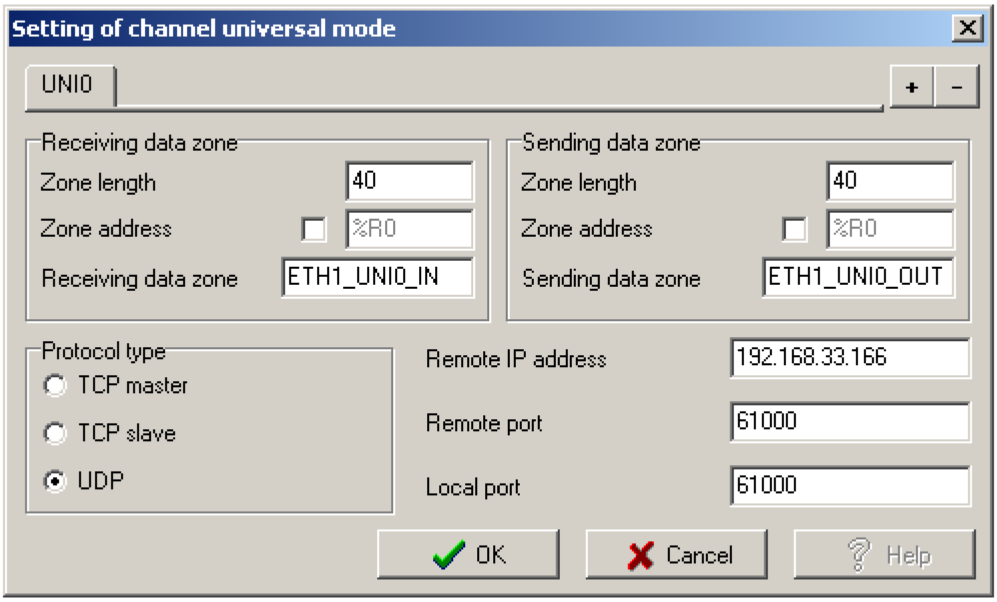
Minimal idle time between sent messages (count of bytes)

The minimum idle period on the line corresponding to the number of sent bytes which is defined between two sent messages. This parameter ensures that between two sent message there will be the idle period on the line minimally of this length kept.

2.5.1.2. Ethernet interface

By pressing the ☑ button on the line of Ethernet interface UNI mode a window *Setting of channel universal mode* will appear (fig.2.5.3).

Within the ETH1 channel (Ethernet on the central unit) there can be realized right one connection at **UNI** mode. Within the ETH2 channel (Ethernet on the communication module SC-710x) there can be realized up to 8 independent connections at UNI mode.



Setting of channel universal mode

UNI0

Receiving data zone
Zone length: 40
Zone address: □ %R0
Receiving data zone: ETH1_UNI0_IN

Sending data zone
Zone length: 40
Zone address: □ %R0
Sending data zone: ETH1_UNI0_OUT

Protocol type
○ TCP master
○ TCP slave
● UDP

Remote IP address: 192.168.33.166
Remote port: 61000
Local port: 61000

Fig.2.5.3 UNI mode parameters setting for the Ethernet interface

Settings meaning and possibilities of individual items for one connection are as follows:

Receiving data zone

Zone length - Length of data section of a receiving zone of the channel.
Zone address - Index of an R register in which the receiving zone starts. If the check box before the entry is enabled, the entry can be changed, otherwise it is generated automatically.
Receiving data zone - Symbolic name of the receiving zone.

Sending data zone

Zone length - Length of data section of a sending zone of the channel.
Zone address - Index of register R in which the sending zone starts. If the check box before the entry is enabled, the entry can be changed, otherwise it is generated automatically.
Sending data zone - Symbolic name of the sending zone.

Protocol type

TCP master - The data transmission via TCP protocol. PLC actively create and keep the connection with the opposite station. The opposite station should be passive (TCP slave) and wait for the connection establishment.
TCP slave - The data transmission via TCP protocol. PLC do not create the connection with the opposite station but wait till this station undertake the task itself. The opposite station, therefore, should be TCP master.
UDP - The data transmission via UDP protocol.

Remote IP address

IP address of the opposite station which the connection is established with.

This entry is opting out in case of TCP slave. PLC waits for the invitation from TCP master irrespective of its IP address. (see chapter 2.5.3.5.).

Remote port

Input port of the opposite station which the connection is established with.

This entry is opting out in case of TCP slave. PLC waits for the invitation from TCP master irrespective of its port (see chapter 2.5.3.5.).

Local port

Input PLC port - port number where PLC receives messages from the opposite station. Any number can be entered apart from 61680 - 61699 can be entered that are dedicated for the system use only.

2.5.2. Data structure

The operation of the serial channel in the **UNI** mode is performed via communication zones in the scratchpad memory that are declared using a fill-in form.

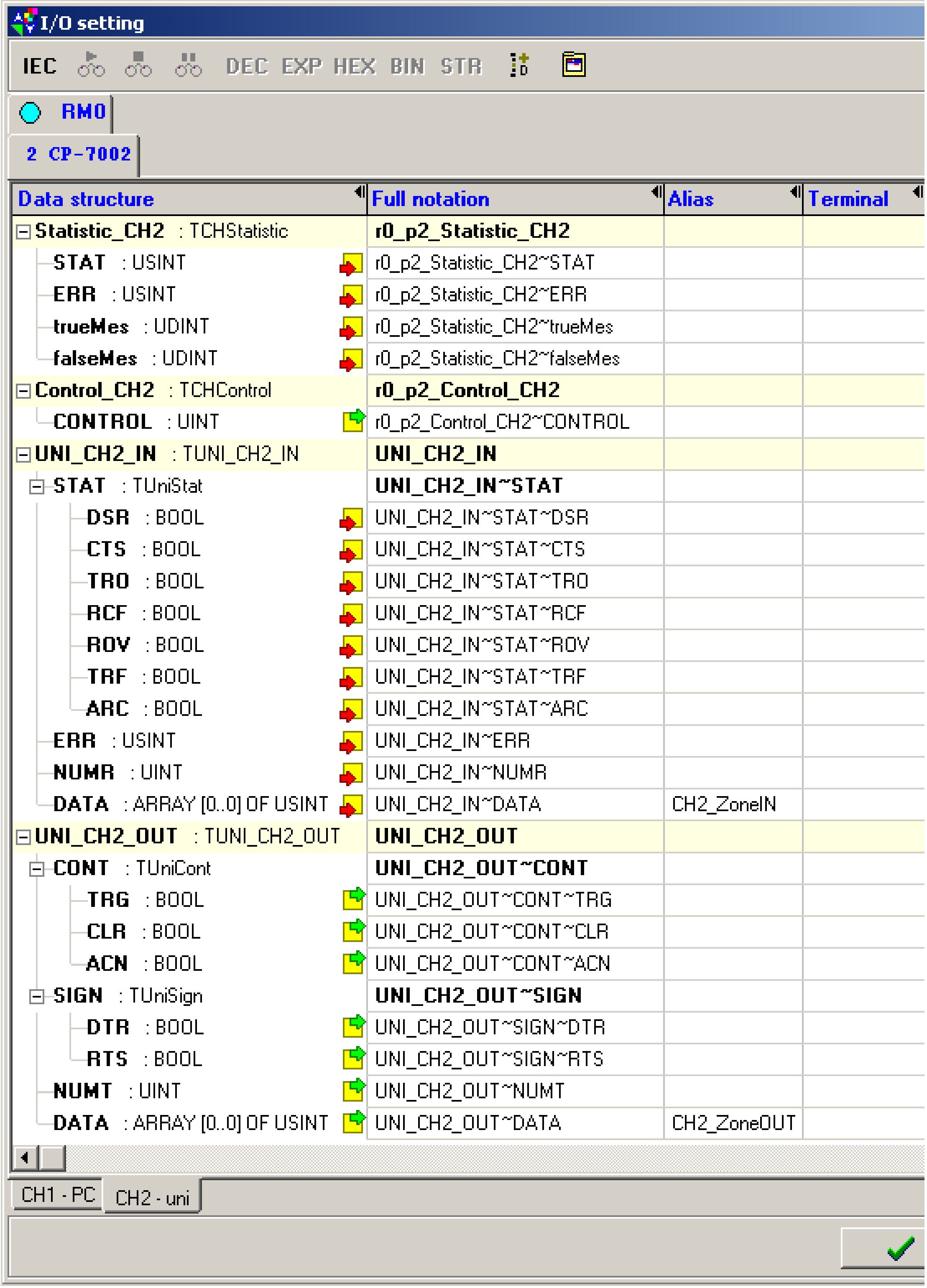
The organization of diagnostic data and communication zones

The communication channel in the **UNI** mode releases diagnostic data of the line status, received and sent data and the status of communication. Data are stored in the scratchpad and are easily accessible through the panel *I/O setting*, available via the icon IO (fig..2.5.4).

Data have assigned symbolic names which begin with the rack and position number. In the column *Full notation*, a concrete symbolic name for the given item is always specified.

If we want to use data in the user program, we use either this symbolic name or we enter our symbolic name, which we then can use, in the column *Alias*. In no case do we use absolute operands, since they can alter after a new compilation of the user program.

Data to be sent are entered by the user into the sending zone. Received data are in the receiving zone. Before data processing, it is necessary to check the status value.



Data structure	Full notation	Alias	Terminal
⊟ Statistic_CH2 : TCHStatistic	r0_p2_Statistic_CH2		
STAT : USINT	r0_p2_Statistic_CH2~STAT		
ERR : USINT	r0_p2_Statistic_CH2~ERR		
trueMes : UDINT	r0_p2_Statistic_CH2~trueMes		
falseMes : UDINT	r0_p2_Statistic_CH2~falseMes		
⊟ Control_CH2 : TCHControl	r0_p2_Control_CH2		
CONTROL : UINT	r0_p2_Control_CH2~CONTROL		
⊟ UNI_CH2_IN : TUNI_CH2_IN	UNI_CH2_IN		
⊟ STAT : TUniStat	UNI_CH2_IN~STAT		
DSR : BOOL	UNI_CH2_IN~STAT~DSR		
CTS : BOOL	UNI_CH2_IN~STAT~CTS		
TRO : BOOL	UNI_CH2_IN~STAT~TRO		
RCF : BOOL	UNI_CH2_IN~STAT~RCF		
ROV : BOOL	UNI_CH2_IN~STAT~ROV		
TRF : BOOL	UNI_CH2_IN~STAT~TRF		
ARC : BOOL	UNI_CH2_IN~STAT~ARC		
ERR : USINT	UNI_CH2_IN~ERR		
NUMR : UINT	UNI_CH2_IN~NUMR		
DATA : ARRAY [0..0] OF USINT	UNI_CH2_IN~DATA	CH2_ZoneIN	
⊟ UNI_CH2_OUT : TUNI_CH2_OUT	UNI_CH2_OUT		
⊟ CONT : TUniCont	UNI_CH2_OUT~CONT		
TRG : BOOL	UNI_CH2_OUT~CONT~TRG		
CLR : BOOL	UNI_CH2_OUT~CONT~CLR		
ACN : BOOL	UNI_CH2_OUT~CONT~ACN		
⊟ SIGN : TUniSign	UNI_CH2_OUT~SIGN		
DTR : BOOL	UNI_CH2_OUT~SIGN~DTR		
RTS : BOOL	UNI_CH2_OUT~SIGN~RTS		
NUMT : UINT	UNI_CH2_OUT~NUMT		
DATA : ARRAY [0..0] OF USINT	UNI_CH2_OUT~DATA	CH2_ZoneOUT	

*Fig.2.5.4 Serial channel data in the **UNI** mode*

Network diagnostics:
Input data *rx_py_Statistic_XXX* (*TCHStatistic* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Stat - communication status (8-times bool type)

	X	X	X	X	X	X	X	CON
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

CON - state of the TCP connection (CP-7004 and FOXTROT from sw version 3.7 only)
- 0 - disconnected
- 1 - connected

Err - communication error (see chapter 2.5.6) (usint type)
trueMes - number of valid network communications (udint type)
falseMes - number of invalid network communications (udint type)

Output data *rx_py_Control_XXX* (*TCHControl* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Control - communication control (16-times bool type)

	X	X	X	X	X	X	X	RES
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

	X	X	X	X	X	X	X	X
bit	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*

RES - reset TCP connection (CP-7004 and FOXTROT from sw version 3.7 only)
- 0 - without change
- 1 - reset connection

Communication zones:
Input data *UNI_XXX_IN* (*TUNI_ XXX_IN* structure):

(XXX - communication channel designation)

Stat - sending and receiving status (8-times bool type)

	ARC	TRF	ROV	RCF	TRO	X	CTS	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

ARC - reception alternation
- the bit changes with newly received message

TRF - transmission in progress, the entry of the next message will be accepted after sending only (log.1)

ROV - overflow (log.1)
- the message received is longer that the receiving zone reserved

RCF - the receiving stacks are full, the message already received will be lost (log.1)

TRO - the sending stacks are full, the entry of the next message will be invalid (log.1)

CTS - CTS signal status (transmission readiness)

Err - reception error (see chapter 2.5.6.) (usint type)
NumR - number of received bytes (uint type)
Data [x] - received message (usint type array element)

Output data *UNI_XXX_OUT* (*TUNI_ XXX_OUT* structure):

(XXX - communication channel designation)

Cont - control of sending and receiving (8-times bool type)

	ACN	CLR	TRG	X	X	X	X	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

ACN - control alternation
- during the bit alternation the value of other CONT bits is accepted

CLR - sending and receiving stacks clearing (log.1)
TRG - message transmission triggering (log.1)

Sign - modem signals control (8-times bool type)

	X	X	X	X	X	X	RTS	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

RTS - RTS signal control

NumT - number of sent bytes (uint type)
Data [x] - sent message (usint type array element)

2.5.3. Basic conditions for data interchange

Passing on messages meant for sending from the user program to the internal broadcasting stack of the communication channel and received messages acceptance from the internal broadcasting stack of the communication channel to the user program is proceed during the I/O scan with ensured data time consistency only (i. e. identical "age" of all message bytes). The exception is a message broadcasting to the serial channel in a case when a requirement on the broadcasting was set at a moment when the broadcasting stack of the channel was not available yet. The new message is collected from the scratchpad immediately after the channel stack is available, it means even during processing of the user program. (see chapter 2.5.3.1.).

The serial channel has a receiving stack and a broadcasting stack, each of 512 bytes size. The Ethernet interface enables to send and receive data of up to 1350 bytes size.

2.5.3.1. Message broadcasting

Message broadcasting procedure

We write a message to the sending zone of the serial channel, then we set the TRG bit at variable Cont to log.1 and change the value of the ACN bit alternation. Broadcasting of the next message is activated by changing the ACN alternation bit. (the TRG bit is still log.1). If we want just to clear stacks and not to broadcast, we have to set the TRG bit to zero before the ACN bit alternation (chapter 2.5.3.3.).

During the message broadcasting through the serial channel is the TRF bit at the variable Stat set to log. 1.

The Ethernet interface is equipped with the broadcasting stack with a sufficient capacity where the whole message is transmitted during the I/O scan to. In the following I/O scan, there can be without any restriction transmitted another message to be sent. To ensure the connection stability it is necessary to communicate at a period shorter then 2 sec.

Sending of more messages consecutively

Before operating the serial channel from the user program it is necessary to realize how long the transmission of one message and one cycle period of the user program will take. The TRF at variable Stat is used for this check. If it has the value of log.0, the sending buffer is empty and broadcasting will be commenced at the I/O scan after entering of a new message (see previous paragraph).

Fully duplex communication can cause channel overload

If the communication is fully duplex (simultaneous independent operation of the receiver and transmitter) it is necessary to ensure by proper timing that the messages are not sent to the serial channel more often than the channel is able to broadcast them further. The full sending stack is indicated by the TRF bit at variable Stat by log.1.

If we write another message during sending that is to be sent, too, this message will not be taken-over from the scratchpad immediately but after the previous message had been sent. This status is indicated by the TRO bit at variable Stat which is set to log.1 and is set to zero at the moment of re-copying of the message from the scratchpad into the free sending buffer.

For the time when the TRO bit is set to log.1, we must not use the content of the sending zone of the serial channel, otherwise the message which is already stored at the scratchpad and is waiting to be sent, could be lost.

This state is necessary to be considered as marginal and it is desirable that this does not happen at all. This state does not occur within the Ethernet interface.

The reception restriction during the broadcasting

If the communication protocol is of type query - response, this problem is avoided since sending of a next message waits for reception of the response to the previous message. Possible hazards at marginal states of communication can be eliminated by restriction of reception during broadcasting by enabling option *Data receiver off on transmit*. This problem does not occur within the Ethernet interface.

2.5.3.2. Message receiving

Message reception procedure

If the communication channel receives a message, it passes the message during the I/O scan to the receiving zone in the PLC scratchpad. The ARC bit at variable Stat changes its value (alternates). If the message received is larger than the size of the receiving zone in the scratchpad, the ROV bit at variable Stat is set to log.1.

Reception of more messages consecutively

Before operating the serial channel via the user program it is necessary to realize how long the reception of one message and one cycle period of the user program will take. The receiving stack has a buffering function so it is possible to receive the next message before the previous message is passed on to the scratchpad. If the stack is full, the RCF bit at variable Stat has the value of log.1. The serial channel will receive the next message over the previous message which will be irretrievably lost. After passing of one message, the stack becomes available for the next reception and the RCF bit changes to log.0.

The Ethernet interface is equipped with the receiving stack with the sufficient capacity where during each scan cycle is the whole message transmitted from, providing it is received. In the following scan the next received message is transmitted, tec. The RFC bit is not set.

2.5.3.3. Resetting of sending and receiving stacks

By setting the CLR bit at variable Cont to log.1 we clear sending and receiving stacks. All messages stored there will be lost. The settings of the channel remains unchanged. The bit must be set to zero in the following cycle again. This command can be combined with the command for sending, stack clearing will be always performed first. The ACN bit at the variable Cont alternation is the condition for accepting of this command.

2.5.3.4. Modem signals control

The serial channel enables the RTS control in four various modes and monitor the CTS signal. The modem signal control mode is set in the set-up panel in the Mosaic development environment for each signal separately.

RS-485 Interface

The RS-485 interface requires a RTS modem signal treatment. This interface is semi-duplex which means that sending or reception can be performed on one line only. Switching between these modes ensures the RTS signal. That means that in case the RS-485 interface is used, such a communication protocol must be used which ensures that communication will take place only at one direction in one moment (query - response principle). It is valid that during sending the RTS = log.0 and during reception the RTS = log.1. The best way is to use RTS automatic control mode (mode *auto value*).

Permanent setting of RTS to a fixed value

If we want the RTS signal to be always on the same pre-defined level, we will use option *permanently value 0* or *permanently value 1*.

RTS control from the PLC user program

If we want the RTS signal to be controlled directly from the PLC user program, we use option *depends on SIGN.1 (from program)*. From the moment of the user program commencement, the signal level is determined by the RTS bit value at the SIGN byte of the corresponding sending zone.

Automatic change of RTS during broadcasting

If we want the RTS signal to be changed automatically during the message sending period, for example owing to the use of the RS-485 interface, we use option *auto value*.

The RTS level will be in the idle condition and in log.1 during reception. Immediately before the broadcasting starts it changes to log.0 and returns to the original condition immediately after sending the last message character.

Change of RTS during broadcasting, the broadcasting is conditioned by CTS

If we want the RTS signal to be changed automatically during message sending and if we want to condition the message sending by a certain CTS signal, for example owing to modem connection, we will use option *auto value with CTS detection.*

If we want to connect a modem to a serial channel, we usually need to induce broadcasting after RTS signal is acknowledged by the CTS signal that reports that the modem is ready to broadcast. This option is used to undertake that.

The RTS level will be in the idle condition and in log.1 during the reception Immediately before the broadcasting commencement it changes to reverse value but sending is initiated only after this level is detected within the CTS signal. The RTS level returns back immediately after sending the last character of the message.

2.5.3.5. Ethernet interface parameters setup and control

UDP protocol

In case of UDP protocol the connection is created so that parameters for both stations are defined according to the following example:

	PLC	remote station
protocol	UDP	UDP
home IP address	192.168.33.160	192.168.33.166
local port	61000	61001
remote IP address	192.168.33.166	192.168.33.160
remote port	61001	61000

Within this setting both stations pursue the mutual connection establishment. Then data are sent via UDP protocol. The local and also the remote port can have identical values.

If the local port number in the PLC is set to 0 then the port number is assigned automatically every time when port is open.

If a connection of more remote stations is required (obviously only one of them can communicate at a given moment) then parameters are defined as follows:

	PLC	remote station 1	remote station 2
protocol	UDP	UDP	UDP
home IP address	192.168.33.160	192.168.33.166	192.168.33.169
local port	61000	61001	61002
remote IP address	0.0.0.0	192.168.33.160	192.168.33.160
remote port	0	61000	61000

Within this setting the PLC waits till some of remote stations establish the connection. Then data are sent via UDP protocol. If the remote station cease communication then after approx. 2 sec is the connection cancelled and another connection can be established by another station. Local and also remote ports can have identical values. To ensure the connection stability it is necessary to communicate within the period shorter than 2 sec.

TCP protocol

In case of TCP protocol it must be resolved which station will actively create and maintain the connection. If we want the PLC to establish the connection (TCP master, we define parameters according to the following example:

	PLC	remote station
protocol	TCP master	TCP slave
home IP address	192.168.33.160	192.168.33.166
local port	61000	61001
remote IP address	192.168.33.166	0.0.0.0
remote port	61001	0

Within this setting the PLC immediately starts to establish the connection with the remote station. It must wait till the connection is established. If both of them would try to establish the connection simultaneously then the connection will not be established at all. After the connection establishment, data are sent via TCP protocol. Local and also remote ports can have identical values.

If a local port number in TCP master PLC mode is set to 0, then port number is assigned automatically with each opening. Number of a local port must be uniquely determined in TCP slave mode.

If a connection of more remote stations is required (obviously only one of them can communicate at a given moment) then parameters are defined as follows:

	PLC	remote station 1	remote station 2
protocol	TCP slave	TCP master	TCP master
home IP address	192.168.33.160	192.168.33.166	192.168.33.169
local port	61000	61001	61002
remote IP address	0.0.0.0	192.168.33.160	192.168.33.160
remote port	0	61000	61000

Within this setting the PLC waits till some of remote stations establish the connection. Then data are sent via TCP protocol. If the remote station cease communication then after several minutes is the connection cancelled and another connection can be established by another station. Local and also remote ports can have identical values.

If RES bit is set in *Control* variable channel diagnostics, connection reset will be done. Reset means connection closure. If PLC is set as master, immediately it newly opens connection. If PLC is set as slave it expects connection with remote station. This function is implemented at all PLC FOXTROT central units and in CP-7004 central unit of TC700 commencing software version 3.7.

2.5.4. Communication functions and their options

Communication functions of the general user channel are used primarily to intercept and pass on the message safely and at once. The tendency is in particular to eliminate problems with received message disruption, with connection of messages following each other and last but not least, in case of network creation, the elimination of messages which are addressed to other participants. For this purpose functions of message start and end detection, message length and station address detection are used. These and other functions also perform further checks of message frame and data validity so that only verified data are passed on to the user program. Individual services of the general user channel and their functions are described in detail in the following chapters.

These functions are used for communication through serial channels only. Within the Ethernet interface are these functions not used because data are always transmitted altogether just as they are transmitted in the data zone of UDP or TCP protocols.

2.5.4.1. Message initial character

Message initial character detection

If each message begins with a certain defined fixed character, we can clearly appoint the start of the message by the initial character detection.

This function is activated by setting an option *Start delimiter* | *Detection*. The value of the initial character is carried by a *Character code* item.

The initial character detection is active only after closing the previous message by any of other function (terminal character detection, idle condition on the line detection, exceeding of maximum length detection, etc.). Consequently, it is enabled that data carried in the message can be of any value, thus, also of a value identical with the value of the initial character. It results from the above mentioned that it is desirable one or more function clearly identifying the message end to be used.

Example:

Initialisation: Start delimiter - Detection, Character code = $48
Minimal idle time between received messages = 3
(the message end is appointed by the line idle condition)
Received message from the line: $48 $01 $02 $03 $04
Message passed on to the system: length - 4, data - $01 $02 $03 $04

Sending of message initial character

By setting an option *Start delimiter* | *Send* at the initialisation panel, we achieve that before each message ready to be send the initial character is added, the value of which is specified by item *Character code*.

Example:

Initialisation: Start delimiter - Send, Character code = $48
Message taken over from the system: length - 4, data - $01 $02 $03 $04
Message sent to the line: $48 $01 $02 $03 $04

2.5.4.2. Message terminal character

Message terminal character detection

If each message ends with a certain defined fixed character or two characters, we can appoint the end of the message by the terminal character detection.

This function is activated by setting an option *End delimiter* | *Detection*. The value of the initial character is carried by a *Character code* item.

If the terminal character is represented by a pair of characters, we thick an option *Two characters* and set their value in the *Character code* item.

If message length is known (using length detection function), the detection of the terminal character is active only after reception of the given number of bytes. Consequently, it is enabled that data carried in the message can be of any value, thus, also of a value identical with the value of the terminal character.

If message length is unknown, the terminal character detection is active all the time and each byte (or more precisely a pair of bytes) is considered to be the terminal character, the value of which is identical with the value specified at the initialisation panel.

Example

Initialisation: End delimiter - Detection, Two characters, Character code = $0A, $0D
Minimal idle time between received messages = 3
Received message from the line: $41 $42 $43 $44 $0A $0D
Message passed to the system: length - 4, data - $41 $42 $43 $44

Sending of the message terminal character

By setting an option *End delimiter* | *Send* at the initialisation panel we achieve that to the end of each message ready to be sent the terminal character is added, the value of which is specified by the *Character code* item. If it is a pair of items, we thick an option *Two characters*.

Example

Initialisation: End delimiter - Send, Two characters, Character code = $0A, $0D
Message taken over from system: length - 4, data - $41 $42 $43 $44
Message sent to the line: $41 $42 $43 $44 $0A $0D

2.5.4.3. Data free acknowledgement

Acknowledgement character detection

Some protocols use for connection testing or for confirmation of a message reception a special character or a combination of two characters. Usually, it concerns protocols with the defined initial character, which is different from the acknowledgement character.

This function is initiated by an option *Confirm message without data* | *Detection*. The value of the terminal character is carried by the *Character code* item. If the acknowledgement character is represented by a pair of characters, the item *Two characters* is enabled.

The acknowledgement character detection is active only after closing the previous message. Since the acknowledgement message is data free, then after successful detection the message is closed and information is passed on to the system as the message with zero length.

Example:

Initialisation: Terminal character - Detection, Character code = $1B
Minimal idle time between received messages = 3
Received message from the line: $1B
Message passed on to the system: length - 0, data - /

Sending of acknowledgement character

By setting an option *Confirm message without data* | *Send* at the initialisation panel, we achieve that the passing on of zero value message for broadcasting is interpreted as the sending of data free confirmation. The value of the acknowledgement character is carried by the *Character code* item. If it is a pair of characters, we thick an option *Two characters*.

Example:

Initialisation: Terminal character - Send, Character code = $1B
Message taken over from the system: length - 0, data - /
Message sent to the line: $1B

2.5.4.4. Station address

Station address detection

If the serial channel is connected to a network with more participants, it is necessary to differentiate addresses of these participants. The protocol used carries the address value always on the defined position in the message. The station address detection allows selecting and passing on only such messages to the system which are assessed for the system.

This function is activated by an option *PLC address* | *Detection on receive*. The position in the message received on which the station address, which the message is designated for, is carried, is determined by the value of this option. The position in the message is numbered from 0. The station address value is entered in the item *PLC address*.

This function compares the byte value on the position given by settings with the value of the specified station address. If values are identical, the message is cancelled and it is not passed on to the system. However, other check and detection functions work normally to ensure correct reception of the following message. The detected address is not passed to the system.

Example:

Initialisation: PLC address = 2
Start delimiter - Detection, Character code = $68,
End delimiter - Detection, Character code = $16,
PLC address - Detection on receive = 1
Minimal idle time between received messages = 3
(start and end of message is determined by defined characters)
Received message from the line: $68 $02 $00 $01 $02 $03 $04 $16
Message passed on to the system: length - 5, data - $00 $01 $02 $03 $04

Sending of station address

By setting an option *PLC address* | *Write on transmit,* we achieve that the station address specified at the initialisation panel in the *PLC address* item is entered to the position given by the value of this option. The position in the message is numbered from 0 and includes all sent bytes, even those added by serial channel services. If we enter the address entry to the position between data passed on from the system, then the address is inserted into this position (see example).

Example:

Initialisation: PLC address = 2
Start delimiter - Send, Character code = $68,
End delimiter - Send, Character code = $16,
PLC address - Write on transmit = 2
Message taken over from the system: length - 5, data - $00 $01 $02 $03 $04
Message sent to the line: $68 $00 $02 $01 $02 $03 $04 $16

2.5.4.5. Check sum

Detection and evaluation of the message data check sum

Check sum is an eight-bit value received by adding up identified message data regardless of transfers to higher binary orders (from mathematical point of view, it is a remainder of integral division of the whole sum by 256). This value is carried at the end of the message contiguously before the terminal character, if it exists, thus, data validity can be checked on the receiving side.

This function is activated by setting an option *Check sum* | *Check on receive*. The position in the received message which the check sum is calculated from, is appointed by the item *Pos. first char. of CHS*. The position in the message is numbered from 0.

This function also calculates the check sum of all data beginning with the given position and the result is compared with the value of the last message byte before the terminal character, if it exists. If they are identical, the message is passed to the system. If they are not identical, error message is sent to the system.

Check sum function requires the message to have the data length defined either directly by a value carried in the message or by maximum number of received bytes or the message to be finished with the terminal character.

Example:

Initialisation: Start delimiter - Detection, Character code = $68,
End delimiter - Detection, Character code = $16,
Check sum - Check on receive, Pos. first char. of CHS = 1
Minimal idle time between received messages = 3
(start and end of the message is determined by defined characters)

Received message from the line:	$68 $01 $02 $03 $04 $0A $16
Message passed to the system:	length - 4, data - $01 $02 $03 $04

Calculation and sending of message data check sum

By setting *Check sum* | *Calculation on transmit* we achieve that the check sum of all data beginning by the position given by the *Pos. first char. of CHS* item is calculated and the result is added to the message end before the terminal character, it exists.

Example:

Initialisation: Start delimiter - Send, Character code = $68,
End delimiter - Send, Character code = $16,
Check sum - Calculation on transmit, Pos. first char. of CHS = 1

Message taken over from the system:	length - 4, data - $01 $02 $03 $04
Message sent to the line:	$68 $01 $02 $03 $04 $0A $16

2.5.4.6. Data length

Message data length detection

Data length detection enables to use data on the data length carried in the message.

This function is initiated by setting an option *Message length* | *Detection on receive*. The position in the received message where the data length is carried on, is determined by the item *Position of message length*. The position in the message is numbered from 0.

The byte value on the given position is considered as the number of data bytes following immediately afterwards. After this count is used up, the message is closed or the check sum and the terminal character are expected providing that these functions are activated. The check sum and the terminal character are not included in information on data number.

Example:

Initialisation:	Start delimiter - Detection, Character code = $68, End delimiter - Detection, Character code = $16, Message length - Detection on receive, Pos. of message length = 1 Minimal idle time between received messages = 3 (start and end of message is determined by defined characters)	
Received message from the line:		$68 $04 $01 $02 $03 $04 $16
Message passed to the system:		length - 4, data - $01 $02 $03 $04

Message data length sending

By setting an option *Message length* **|** *Write on transmit* we achieve that the data length is entered to the position given by the item *Position of message length.* The position in the message is numbered from 0.

Example:

Initialisation:	Start delimiter - Send, Character code = $68, End delimiter - Send, Character code = $16, Message length - Write on transmit, Position of message length = 1	
Message taken over from system:		length - 4, data - $01 $02 $03 $04
Message sent to the line:		$68 $04 $01 $02 $03 $04 $16

2.5.4.7. Maximum message length

Definition of a maximum data length serves as a protection against the possibility of congestion of the unit by receiving endless data strings. If no other service defining the message end is activated then the maximum length works as a separator of fixed data number regardless of their value. In connection with other functions it serves as the above mentioned protection element.

This function is always active and the maximum length is determined by the value of the *Maximum length* item in the initialisation panel. If zero value is entered, messages of full length up to the size of the receiving zone in the scratchpad are let through.

Example

Initialisation:	Maximum length = 4 Minimal idle time between received messages = 3 (the message end is determined by the line idle condition)	
Received message from the line:		$01 $02 $03 $04 $05 $06 $07
1st message passed on to the system:		length - 4, data - $01 $02 $03 $04
2nd message passed on to the system:		length - 3, data - $05 $06 $07

2.5.4.8. Idle condition on the line

Detection of the idle condition on the line

This function, similarly to the maximum message length represents an important protection element especially in cases of a communication interruption in the middle of the message receiving. If the idle condition on the line is for a longer time than it is specified, the reception of the previous message is terminated. The message is passed to the system and the start of the message detection functions are activated if they were enabled.

This function is activated at non-zero value of the item *Minimal idle time between received messages* in the initialisation panel.

Free channel mode

If the value of the item *Minimal idle time between received messages* is 0, free channel mode is switched on at the reception. The channel fulfils only the role of a symbol reception from the serial line. At each cycle all bytes, which were received within the previous cycle, are passed on to the system. For majority of applications is this mode unsuitable, mainly because depending on the communication speed and cycle period it tears the received message to several parts that then must be connected together by the user program. This mode also cannot separate two messages arriving one after another.

The non-zero value of the item *Minimal idle time between received messages* specifies the minimum idle condition on the line recalculated to the number of bytes received. As a consequence, the resulting time given by this value will be dependant on the transmission rate. An internal timer with a 100 µs clock cycle is used for checking the idle condition on the line. The calculation of the resultant idle condition on the line in time units is performed according to the following formulas:

$$T = 100 \cdot k_T \quad [\mu s]$$

$$k_T = \left(\frac{110000 \cdot TOR}{BD} + 1 \right) \text{mod} 1$$

where BD transmission rate in Bd
TOR the number of bytes corresponding to the minimum idle condition on the line
k_T the number of cycles of the internal timer rounded down to an integral number
T theoretical minimum idle condition on the line

Considering that the internal timer with the 100 µs cycle is running continuously, we must take into account a possible cycle error caused by the fact that the first timer cycle will not be 100 µs but it will be within the interval of <1, 100> µs depending on the difference between the moment of the reception of the last byte and the moment of the last cycle of the internal cycle. The real minimum idle condition on the line T_l will be within the interval of <(T–99), T>.

The value of the idle condition on the line is necessary to set higher than the maximum possible timeout between the individual message bytes. Otherwise, the message could be torn apart.

Example:

Transmission rate 19.2 kBd, minimum idle condition on the line 3 bytes.
(At the rate of 19.2 kBd, reception of 3 bytes takes 1719 µs.)
Initialisation table: Communication rate = 19 200
Minimal idle time between received messages = 3

$$k_T = \left(\frac{110000 \cdot 3}{19200} + 1 \right) \text{mod} 1 = 18{,}1875 \text{mod} 1 = 18$$

$$T = 100 \cdot 18 = 1800 \quad [\mu s]$$

$$T_l = \langle 1701, 1800 \rangle \quad [\mu s]$$

Idle condition on the line between two sent messages

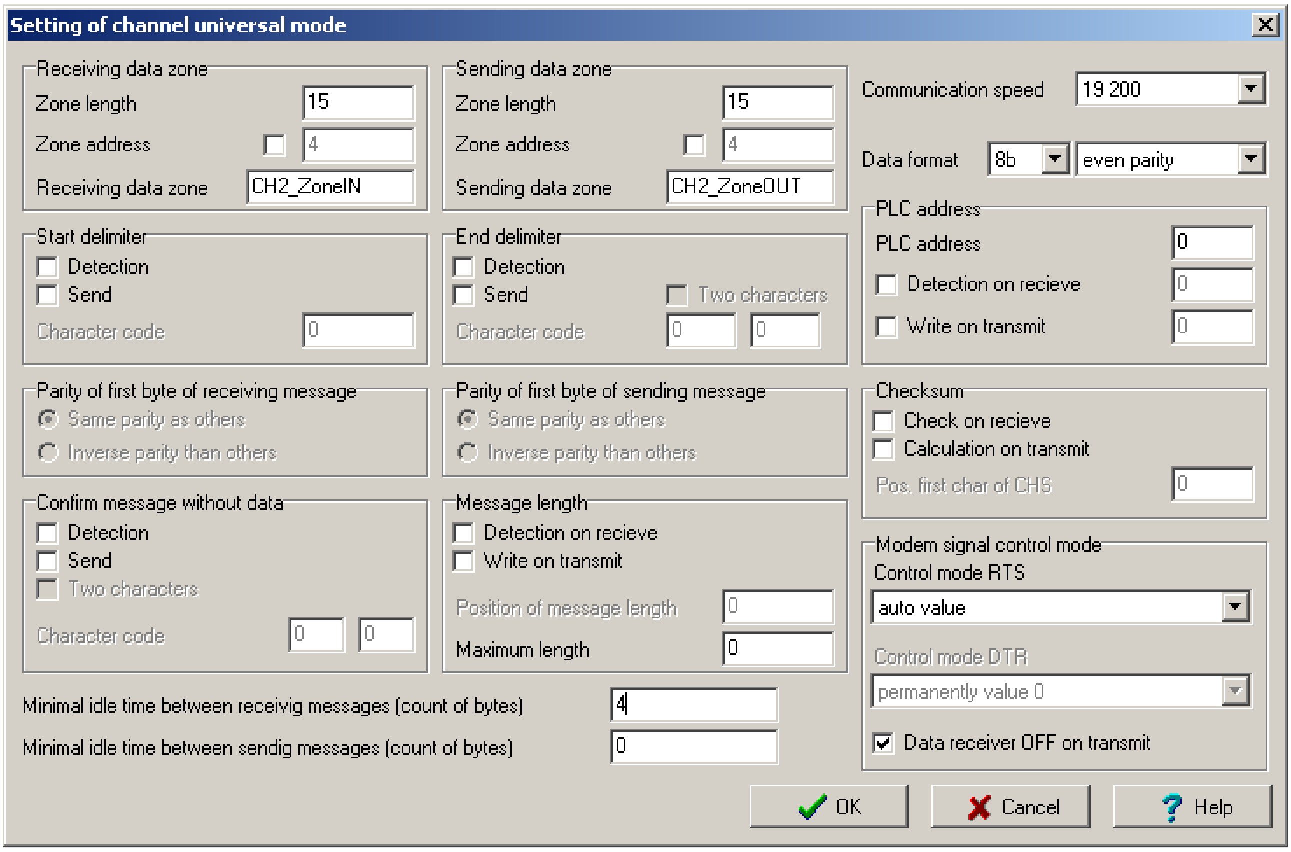
At a non-zero value of the item *Minimal idle time between sent messages* in the initialisation panel, the generation of the idle condition on the line between two sent messages is switched on. In case a message ready to be sent immediately after sending of the previous message is passed from the system to the sending stack, the next message will be sent after the minimum idle condition on the line given by the value specified in the number of sent bytes, elapses. For the calculation of the idle condition on the line in time units, relations given in the previous chapter are applied.

If the value of *Minimal idle time between sent messages* is 0, no idle condition on the line is generated between sent messages.

2.5.5. General devices connection

If received data does not create an integrated message (e.g. ASCII codes of pressed terminal keys), it is possible to leave the channel in free mode (chapter 2.5.4.9.). In absolute majority of cases, however, we require passing on of integrated messages. Then it is necessary to set corresponding function according to the chapter 2.5.2. and chapter 2.5.4. in the initialisation table.

The example in fig. 2.5.5 shows the settings of the serial channel for the reception of integrated messages separated by a four-byte space. The other options are off. So there is a minimum configuration for the reception of integrated messages. The channel must be set to the **UNI** mode.



Setting of channel universal mode

Receiving data zone
Zone length 15
Zone address □ 4
Receiving data zone CH2_ZoneIN

Sending data zone
Zone length 15
Zone address □ 4
Sending data zone CH2_ZoneOUT

Communication speed 19 200
Data format 8b even parity

Start delimiter
□ Detection
□ Send
Character code 0

End delimiter
□ Detection
□ Send □ Two characters
Character code 0 0

PLC address
PLC address 0
□ Detection on recieve 0
□ Write on transmit 0

Parity of first byte of receiving message
● Same parity as others
○ Inverse parity than others

Parity of first byte of sending message
● Same parity as others
○ Inverse parity than others

Checksum
□ Check on recieve
□ Calculation on transmit
Pos. first char of CHS 0

Confirm message without data
□ Detection
□ Send
□ Two characters
Character code 0 0

Message length
□ Detection on recieve
□ Write on transmit
Position of message length 0
Maximum length 0

Modem signal control mode
Control mode RTS
auto value
Control mode DTR
permanently value 0
☑ Data receiver OFF on transmit

Minimal idle time between receivig messages (count of bytes) 4
Minimal idle time between sendig messages (count of bytes) 0

*Fig.2.5.5 **UNI** mode parameters setup according to the example*

2.5.6. Error messages

Operational error messages from the serial channel are displayed in the receiving zone at variable *Err* and in the diagnostics zone at variable *Err*.

$10 Invalid initial character

The value of the message initial character does not correspond to the value specified in the initialisation table.

$11 Parity error

At least one message byte had invalid parity.

$12 Maximum message length exceeded

The message received is longer than the maximum length specified in the initialisation table.

$13 Invalid second byte of the confirmation

The value of the second byte of the confirmation does not correspond to the value specified in the initialisation table.

$14 Invalid second byte of the terminal character

The value of the second byte of the terminal character does not correspond to the value specified in the initialisation table.

$18 Check sum error

The received value of the check sum does not correspond to the value calculated.

$19 Invalid terminal character

The value of the terminal character does not correspond to the value specified in the initialisation table.

$31 Invalid length of sent data

The value of the length of sent data exceeds the size of the sending zone.

$32 Zero length of sent data

The value of the length of sent data has to be non-zero with the exception of sending of the acknowledgement character.

$40 Timeout not met

The time between two received messages was shorter than the value specified in the initialisation table.

2.6. MDB MODE - COMMUNICATION WITH THE MASTER SYSTEM VIA MODBUS PROTOCOL

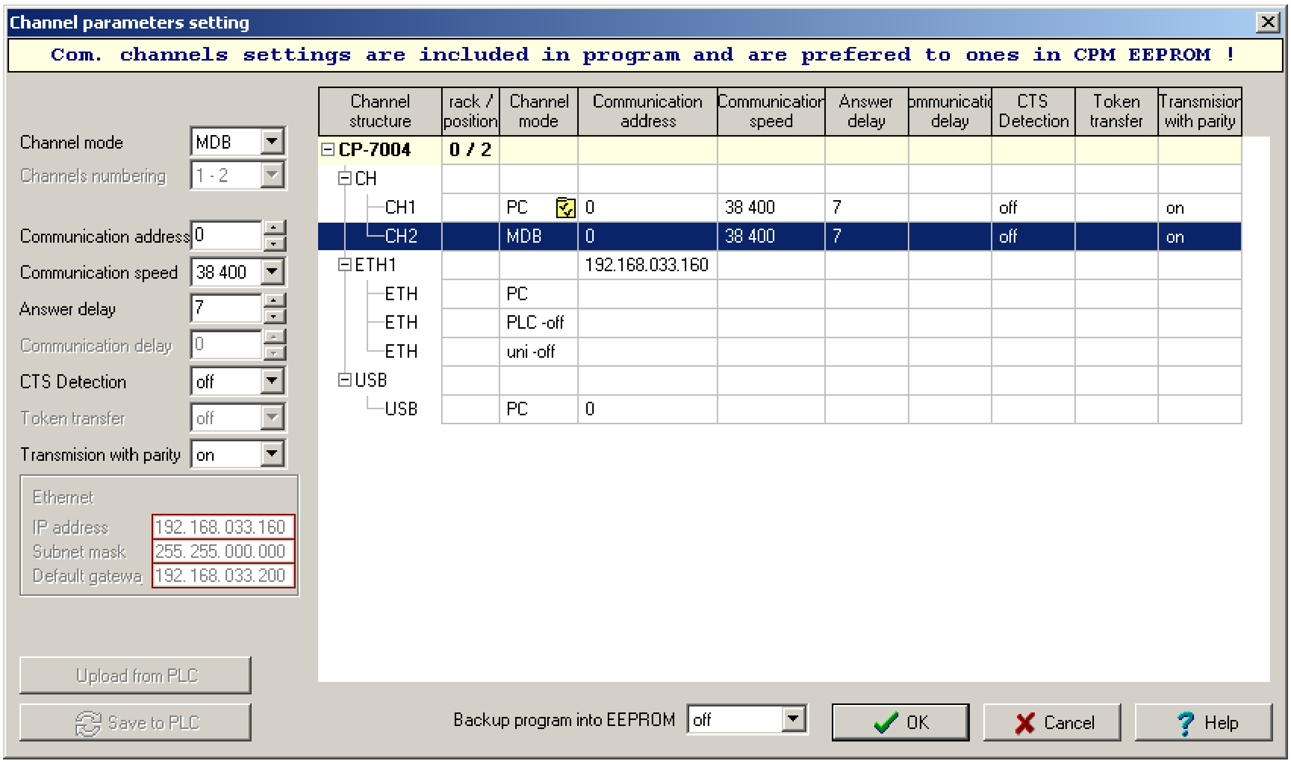
In practice, situations can occur when the communication via the EPSNET network cannot be used. This situation for example can occur when it is necessary to connect an operational panel to the TECOMAT PLC which is not equipped with the EPSNET driver. Another such situation can be the requirement of connection of visualisation software where no EPSNET driver is available and the OPC server for TECOMAT cannot be used.

The above mentioned cases can be resolved by setting the serial channel to the **MDB** mode allowing communication with the master system via the MODBUS protocol. This protocol is well known and wide-spread in the field of industrial communications. The communication is initiated by a master system on the query - response principle. This principle allows connection of a greater number of participants to the master system within the RS-485 interface. Other interfaces allow the connection of one participant (point-to-point connection). TECOMAT in the **MDB** mode behaves as a passive slave participant.

PLC FOXTROT CPUs and CP-7004 CPU of TC700 PLC series support moreover MODBUS TCP and MODBUS UDP protocols on the Ethernet interface commencing firmware version 3.7.

2.6.1. Channel parameters setup

In the Mosaic development environment we select folder *Hw* | *HW Configuration*. In the list of modules we highlight using the mouse the central unit or the communication module, the channel of which we want to set. Then we press the push-button *Settings* or the icon ☑ on the line on the right from the position number. The panel *Channel parameters setting* (fig. 2.6.1) is displayed.



Channel parameters setting

Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

Channel mode	MDB
Channels numbering	1 - 2
Communication address	0
Communication speed	38 400
Answer delay	7
Communication delay	0
CTS Detection	off
Token transfer	off
Transmision with parity	on

Ethernet

IP address	192. 168. 033. 160
Subnet mask	255. 255. 000. 000
Default gatewa	192. 168. 033. 200

Channel structure	rack / position	Channel mode	Communication address	Communicatior speed	Answer delay	ommunicati delay	CTS Detection	Token transfer	Transmisior with parity
⊟ **CP-7004**	**0 / 2**								
⊟ CH									
CH1		PC ☑	0	38 400	7		off		on
CH2		MDB	0	38 400	7		off		on
⊟ ETH1			192.168.033.160						
ETH		PC							
ETH		PLC -off							
ETH		uni -off							
⊟ USB									
USB		PC	0						

Backup program into EEPROM: off

Fig.2.6.1 ***MDB*** *mode setup*

We set the corresponding serial channel to the **MDB** mode, then we set the transmission rate, address, response timeout, CTS signal detection and parity. If we press push-button *Save to PLC*, the settings will be stored in the central unit (not valid for serial channels of communication modules). If we compile the user program, communication settings will become a part of the user program and will be activated in the central unit at the restart of the user program.

No parameters have to be set for **MDB** mode at the Ethernet interface. The interface must have set IP address, IP mask and in case of need gateway IP address.

Moreover, for communication modules we must set the option *Channels numbering* and so assign logic identification to the individual channels (e.g. CH3, CH4). The central unit has the CH1 and CH2 channels fix-assigned.

Address setup

The address selection allows the connection of more TECOMAT and TECOREG systems to one master system (in this case, the RS-485 interface is necessary). It must be ensured here that the connected PLCs as well as the master system always have a different address. Addresses do not need to constitute continuous series.

Transmission rate setup

Serial channels in this mode allow the transmission rate of up to 115.2 kBd. The higher the transmission rate, the shorter the data transmission. Since the communication is running independently of the user program, it affects all cycle phases of the user program evenly. It is valid here that by increasing the transmission rate, the cycle period becomes slightly longer. In table 2.6.1 the cycle period extension in per cent is listed. These values are valid for the maximum channel load (continuous communication). The longer timeouts between the messages, the lower the cycle period extension will be.

Table 2.6.1 Average user program cycle time extension depending on the serial channel transmission rate and the central unit type

Transmission rate	TC650, CP-7001, CP-7002, CP-7003, CP-7005 TC700	FOXTROT, CP-7004 TC700
9.6 kBd	0.3 %	0.1 %
19.2 kBd	0.5 %	0.1 %
38.4 kBd	1.0 %	0.2 %
57.6 kBd	1.5 %	0.3 %
115.2 kBd	3.0 %	0.6 %

Response timeout setup

Response timeout setup allows selection of the minimum period of time that elapses from sending the last byte of the master system message to the beginning of the transmission of the first byte of the response of the questioned TECOMAT system. During this period the master system must get ready for reception. Sometimes, this preparation can take more time, depending on the type of the master system (e.g. on the operation system used on the PC). The usage of some serial interface converters, repeaters, modems and radio modems also requires a longer response timeout. Usually it is a device switching the communication direction (e.g. on the RS-485 interface) or causing a long transmission delay.

Minimum and maximum value of response timeout

A minimum value of response timeout is set. For majority of services, its maximum value depends on the cycle period of the TECOMAT system, since the data are passed on at the I/O scan only to ensure data constancy during user program execution. The minimum timeout response can be fixed within the range of 1 to 99 ms or it can be set to 0 which means the minimum timeout response corresponds to the time required for sending of one byte, it thus depends on the transmission rate according to the table 2.6.2.

Table 2.6.2 Minimum response timeout at set value 0

Transmission rate	Minimum response timeout
0.3 kBd	36.67 ms
0.6 kBd	18.33 ms
1.2 kBd	9.17 ms
2.4 kBd	4.58 ms
4.8 kBd	2.29 ms
9.6 kBd	1.14 ms
14.4 kBd	0.76 ms
19.2 kBd	0.57 ms
28.8 kBd	0.38 ms
38.4 kBd	0.29 ms
57.6 kBd	0.19 ms
115.2 kBd	0.09 ms

CTS signal detection setup

Switching on the CTS signal detection enables to delay the response with the external signal from a modem. The response will be sent after 10 ms following the CTS signal change to the corresponding RTS signal status valid for response sending.

CTS signal detection is primarily intended for the cases of communication via modems. Even with the CTS signal detection, the minimum set timeout response is ensured at the same time. It means that TECOMAT will not start sending the response before the minimum response time elapses even if the CTS signal was already properly set.

The solution of the RTS requirement on advance setting before data transmission

The timeout delay of 10 ms after CTS change detection is intended to "calm down" conditions on the transmission medium before data transmission (e.g. carrier frequency rising time). If the modem does not return the CTS signal but requires a 10 ms timeout between RTS signal setting and data itself, we connect the RTS and CTS signals on the serial channel connector and switch on the CTS signal detection. Setting of the minimum timeout response (previous parameter) does not guarantee the exact moment of RTS signal switching. It is switched at the moment of response creation only, the effect of cycle period approves itself significantly here.

Parity setup

The MODBUS RTU network protocol uses even parity and one stop-bit or none parity and two stop-bits.

2.6.2. Network operation

MODBUS protocol description

Communication via the serial channel in the **MDB** mode corresponds to the behaviour of the MODICON 884 system described in the documentation Modicon Modbus Protocol Reference Guide PI-MBUS-300 Rev. G. In this documentation you can find all the necessary information related to the MODBUS protocol. A list of functions supported by the **MDB** mode is given in table 2.6.3. Systems connected to the MODBUS network can be generally set to one of the two modes - ASCII or RTU. The **MDB** mode supports RTU (Remote Terminal Unit) only. In this mode a minimum idle condition on the line (a pause between individual messages) of at least 3.5 multiple of the time required for sending of one character is presumed.

MDB mode supports MODBUS TCP and MODBUS UDP protocols that correspond to MODBUS RTU protocol by their structure and services, on the Ethernet interface.

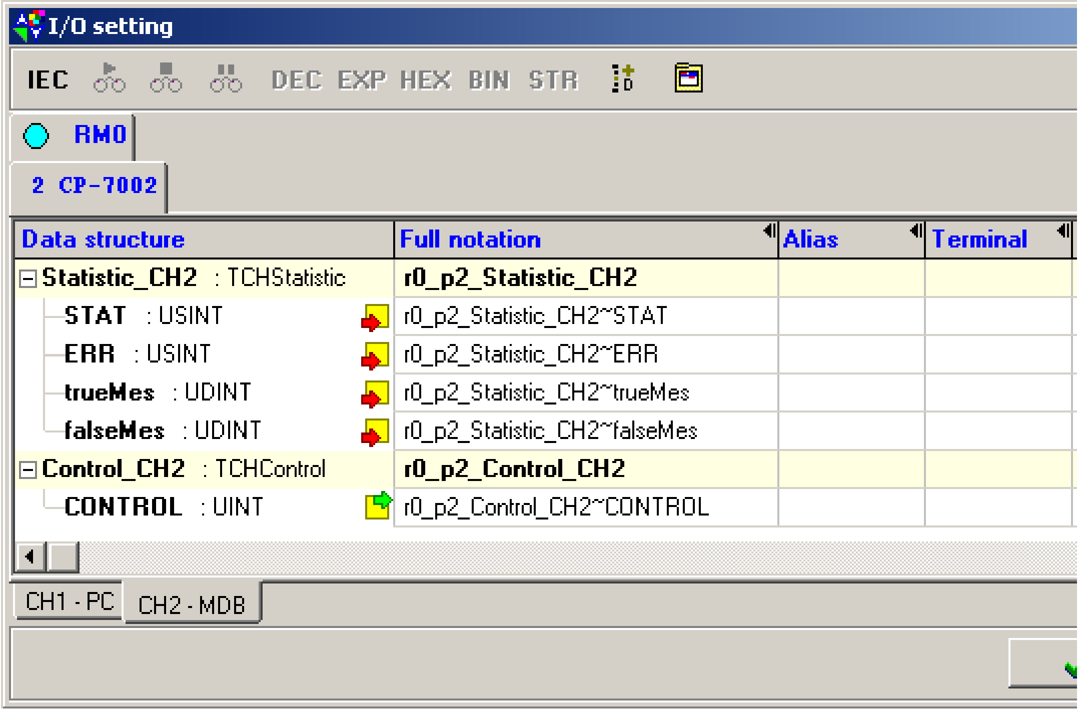
Table 2.6.3 List of functions of the MODBUS protocol supported in the **MDB** mode

Code	Function	Description
01	Read Coil Status	reading of outputs (memory 0X)
02	Read Input Status	reading of inputs (memory 1X)
03	Read Holding Registers	reading of registers (memory 4X)
04	Read Input Registers	reading of input registers (memory 3X)
05	Force Single Coil	setting of one output (memory 0X)
06	Preset Single Register	setting of one register (memory 4X)
07	Read Exception Status	information on controller status
08	Diagnostics	diagnostics functions
15	Force Multiple Coils	setting of outputs (memory 0X)
16	Preset Multiple Registers	setting if holding registers (memory 4X)
17	Report Slave ID	returns controller ID-number

Operation diagnostics

Serial channels in the **MDB** mode, the declaration of which is a part of the user program, release the diagnostic data of a line status. This data are stored in the scratchpad and are easily accessible in the panel *I/O setting* using an icon IO (Fig. 2.6.2).

Diagnostic data have assigned symbolic names that begin with the rack and position number. In the column *Full notation* a concrete symbolic name for the given item is always specified.



Data structure	Full notation	Alias	Terminal
Statistic_CH2 : TCHStatistic	r0_p2_Statistic_CH2		
STAT : USINT	r0_p2_Statistic_CH2~STAT		
ERR : USINT	r0_p2_Statistic_CH2~ERR		
trueMes : UDINT	r0_p2_Statistic_CH2~trueMes		
falseMes : UDINT	r0_p2_Statistic_CH2~falseMes		
Control_CH2 : TCHControl	r0_p2_Control_CH2		
CONTROL : UINT	r0_p2_Control_CH2~CONTROL		

*Fig. 2.6.2 Serial channel data in the **MDB** mode*

If we want to use the data in the user program, we use either this symbolic name or we enter our symbolic name in the column *Alias* which we can use further. In no case do we use absolute operands, since they can alter after a new compilation of the user program.

Network diagnostics:
Input data *rx_py_Statistic_XXX* (*TCHStatistic* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Stat	- communication status (not used for now) (usint type)
Err	- communication error (see chapter 2.6.4) (usint type)
trueMes	- number of valid network communication (udint type)
falseMes	- number of invalid network communication (udint type)

Output data *rx_py_Control_XXX* (*TCHControl* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Control	- communication control (not used for now) (uint type)

2.6.3. MODICON 884 system emulation

User memory assignment

If we want a TECOMAT PLC to behave during communication via the MODBUS protocol similarly as MODICON 884, it has to use the same way of mapping of PLC user memory. Therefore, memory areas of the MODICON 884 system were assigned to corresponding areas of the TECOMAT PLC memory according to table 2.6.4.

Principles of assignment conversion

The first object in the relevant area of the TECOMAT system memory corresponds to the first object of the MODICON system memory. In connection with this, the following facts should be remembered. The numbering of objects in MODICON systems starts with number one unlike in TECOMAT systems where the first object has the index 0. The objects in areas 0xxxx and 1xxxx are numbered as individual bits following each other. If a number of transmitted objects is specified, it specifies the number of transmitted bits. The objects in areas 3xxxx and 4xxxx are numbered as 16-bit words and the number of transmitted objects is thus the number of words following each other. The number of transmitted bytes is in this case the double of the value specified. It should be remembered in this context that TECOMAT has a byte-oriented memory and thus it numbers byte registers. For example, RW0 consists of byte registers R0 and R1, register RW1 consists of registers R1 and R2, register RW2 consists of registers R2 and R3, etc. For communication there are 16-bit words of registers with even numbers available, overlapping each other. The RW number is calculated from specified number 3xxxx or 4xxxx according to the following relation:

(xxxx – 1) x 2

Tab.2.6.4 The conversion table of memory areas between MODICON and TECOMAT systems

MODICON system memory area	Corresponding memory of the TECOMAT system
0XXXX - discrete outputs	**Y - outputs**
00001	Y0.0
00002	Y0.1
00003	Y0.2
...	...
00008	Y0.7
00009	Y1.0
00010	Y1.1
...	...
1XXXX - discrete inputs	**X - inputs**
10001	X0.0
10002	X0.1
10003	X0.2
...	...
10008	X0.7
10009	X1.0
10010	X1.1
...	...
3XXXX - input registers	**SW - system registers**
30001	SW0
30002	SW2
30003	SW4
...	...
30008	SW14
30009	SW16
30010	SW18
...	...
4XXXX - holding registers	**RW - user registers**
40001	RW0
40002	RW2
40003	RW4
...	...
40008	RW14
40009	RW16
40010	RW18
...	...

2.6.4. Error messages

Operational error messages of the serial channel are displayed at variable *Err*. The message received where an error occurred, is ignored and included in the number of false communications (variable *falseMes*). The cause of these errors can be either a high level of interference, invalid parameter settings, incorrectly connected communication line or an error on the side of the master system.

$11 Parity error

Some of message bytes had invalid parity.

$18 CRC error

The CRC value (protection character) of the received message does not correspond to the value calculated.

2.7. PFB MODE - PROFIBUS DP SLAVE STATIONS CONNECTION TO PLC

What is a PROFIBUS DP

PROFIBUS is a data bus used for data transmission in industrials environments among control centres, control systems, etc. From the PROFIBUS FMS bus definition, the PROFIBUS DP was created, used primarily for the connection of remote peripheries (remote I/O), intelligent sensors and actuating mechanisms. The PROFIBUS DP bus is defined by the norm IEC 61158 and is supported by a number of producers of peripheral modules, usually having a high protection class, thus they can be implemented directly in the required technology. The RS-485 interface enables a long-distance transmission with a sufficient resistance against the interference. Data transmission is performed on master station query → slave station response basis. The message protocol has a multiple data protection ensuring that the data successfully transmitted are correct.

PFB mode function

The connection of stations PROFIBUS DP slave to the TECOMAT PLC is accomplished by a standard serial channel in the **PFB** mode fitted with the RS-485 interface. The PFB mode creates PROFIBUS DP master station. The devices connected must be of PROFIBUS DP slave stations supporting communication DP-V0 type (cyclic data exchange - MS0) and no other PROFIBUS DP master station must be available in the network. Communication among stations would require a longer time load of the central unit also in the case where there is no other master in the network which applies to the absolute majority of cases in this field of applications.

The advantage of this solution is a minimum costs on the side of the PLC. A certain disadvantage can be a slower transmission rate - alternatively 187.5 kBd, 93.75 kBd, 19.2 kBd or 9.6 kBd. Since this solution is not designed for the connection to the large PROFIBUS network, these limitations are acceptable.

A PLC in the **PFB** mode establishes communication with the relevant station, performs its parameterisation, configuration, following with data transmission and continuous diagnostics.

To each slave station the *Mode* communication status is maintained (item *Mode* of the station diagnostics zone - see chapter 2.7.3.) informing us about the phase communication is at. If communication is lost, the PLC tries to re-establish communication with the station and after its re-establishing, parameterisation and configuration are repeated. The course of data exchange with the station can be described by a diagram in fig. 2.7.1 which is valid for each slave station separately, independently of the communication status with the other stations.

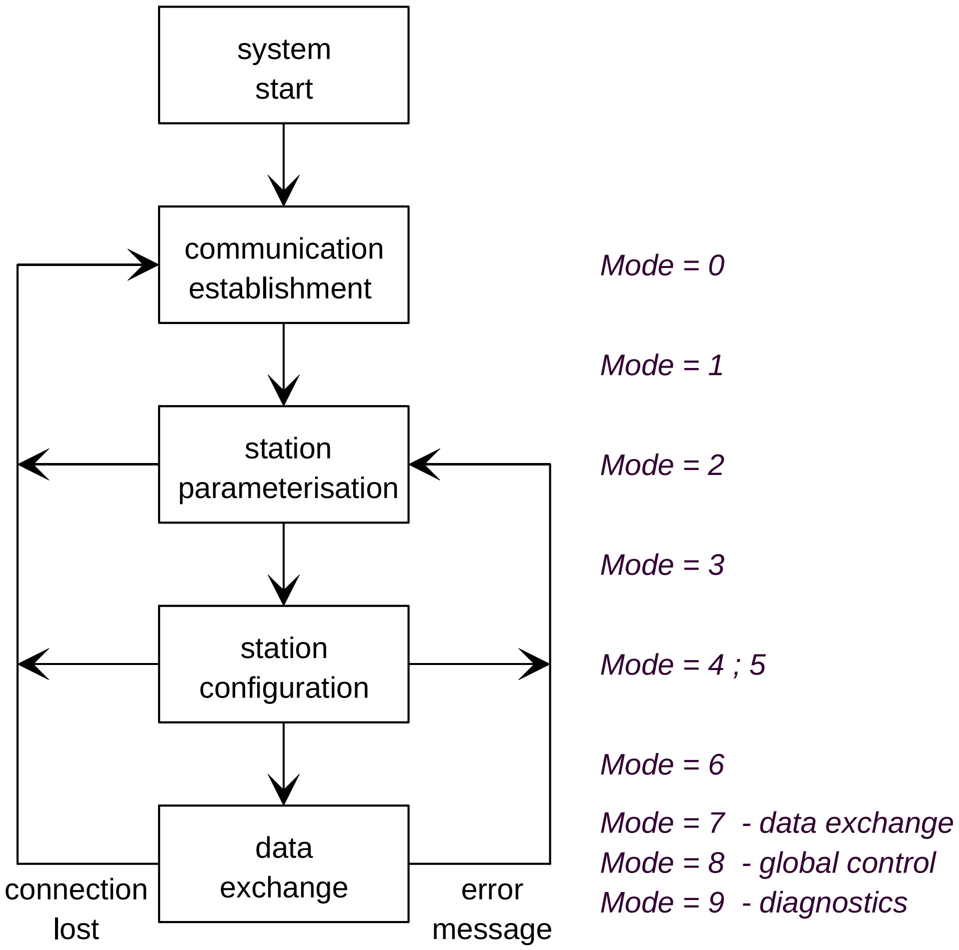


Fig.2.7.1 Diagram of communication states with the slave station

2.7.1. Channel parameters setup

In the Mosaic development environment we select folder *Hw* | *HW Configuration*. In the list of individual modules we highlight the central unit, the channel of which we want to set. Then we press the push-button *Settings* or an icon ☑ on the line on the right from the position number. The panel *Channel parameters setting* (fig. 2.7.2) is displayed.

We set the corresponding serial channel to the **PFB** mode and then also the transmission rate and address.

Address selection

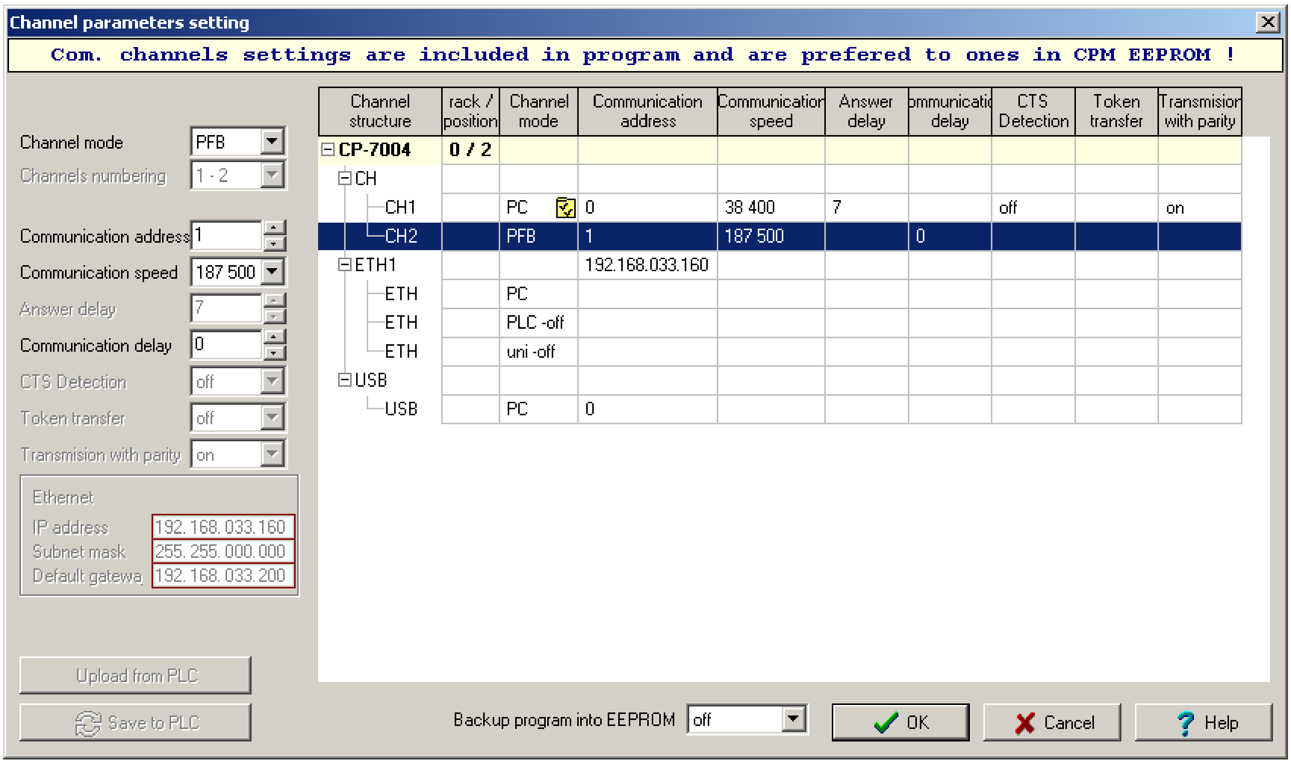
The address set here is the address assigned to the PLC as the master station. Each network participant must have an exclusive address. The addresses do not need to follow-up each other. Address 0 is reserved for special purposes and should not be used.

Transmission rate selection

The PROFIBUS DP bus has defined individual transmission rates that can be used. For communication with TECOMAT PLC's the following values can be used: 187.5 kBd, 93.75 kBd, 19.2 kBd or 9.6 kBd.

Within central units CP-700x TC700 and CP-10xx FOXTROT, is communication on the first two channels CH1 and CH2 performed by the processor of the central unit.

Since communication is running independently of the user program, it affects all cycle phases of the user program evenly. It applies here that by increasing the transmission rate, the cycle period becomes longer. In the table 2.7.1 the cycle time extension in per cent is shown.



Channel parameters setting

Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

Channel mode: PFB
Channels numbering: 1 - 2
Communication address: 1
Communication speed: 187 500
Answer delay: 7
Communication delay: 0
CTS Detection: off
Token transfer: off
Transmision with parity: on

Ethernet
IP address: 192. 168. 033. 160
Subnet mask: 255. 255. 000. 000
Default gatewa: 192. 168. 033. 200

Channel structure	rack / position	Channel mode	Communication address	Communicatior speed	Answer delay	ommunicatic delay	CTS Detection	Token transfer	Transmisior with parity
⊟ CP-7004	0 / 2								
⊟ CH									
CH1		PC	0	38 400	7		off		on
CH2		PFB	1	187 500		0			
⊟ ETH1			192.168.033.160						
ETH		PC							
ETH		PLC -off							
ETH		uni -off							
⊟ USB									
USB		PC	0						

Backup program into EEPROM: off

Fig.2.7.2 ***PFB*** *mode setting*

Table 2.7.1 Average cycle period extension of the user program depending on the serial channel transmission rate

Transmission rate	TC650 CP-7002, CP-7003 TC700	FOXTROT CP-7004 TC700
9.6 kBd	0.3 %	0.1 %
19.2 kBd	0.5 %	0.1 %
93.75 kBd	2.5 %	0.5 %
187.5 kBd	5.0 %	1.0 %

Transmission delay setting

Optional transmission delay serves the resolution of cases when the master system is interconnected with slave systems via modems that are causing a communication delay. Used modems must fulfil conditions for a transparent data transfer. It means data received by the first modem are transmitted by the second modem in the unchanged binary form and time sequence. It must not The message must not get divided (space between bytes must not extend the time necessary for transmitting 33 bits) and all bits in the byte including parity bit must be transmitted.

The transmission delay is set using multiples of 100 ms and can take the value from 0 to 6.0 s. The 0 value signify that the master system anticipates response within the time determined by the slave station description in the GSD file. Values 1 to 60 determine transmission delay from 0.1 s to 6.0 s. Values 61 to 99 set the maximum transmission delay to 6.0 s.

Modems with automatic signal polarity setup

Some modems contain circuits for automatic setup of the transmitted signal polarity. To be set, these modems require idle condition on the line after operation start for about 1 to 2 s. Since the status must be handled right when the modem connects to the master system just after its operation initiation on the serial channel (disconnected cable to the modem, modem power supply failure, etc.), it is necessary the transmission delay of at least 2.0 s is set. After sending a message, the master system anticipate the response for 2.0 s, thus enabling automatic setting of an idle polarity of modem circuits. The following request of the master system will already be sent by the modem correctly.

2.7.2. Network initialisation

Parameters setup in the Mosaic environment

The structure of the entire network PROFIBUS DP connected to the TECOMAT PLC can be created using the Project manager, folder *Hw* | *PLC Network*. We paste the PLC of the current project to the desktop. The PROFIBUS DP slave station is created using an option *Objects* | *Profibus DP* (accessible also using a mouse right button) after which a panel *Device selection* will appear (fig. 2.7.3).

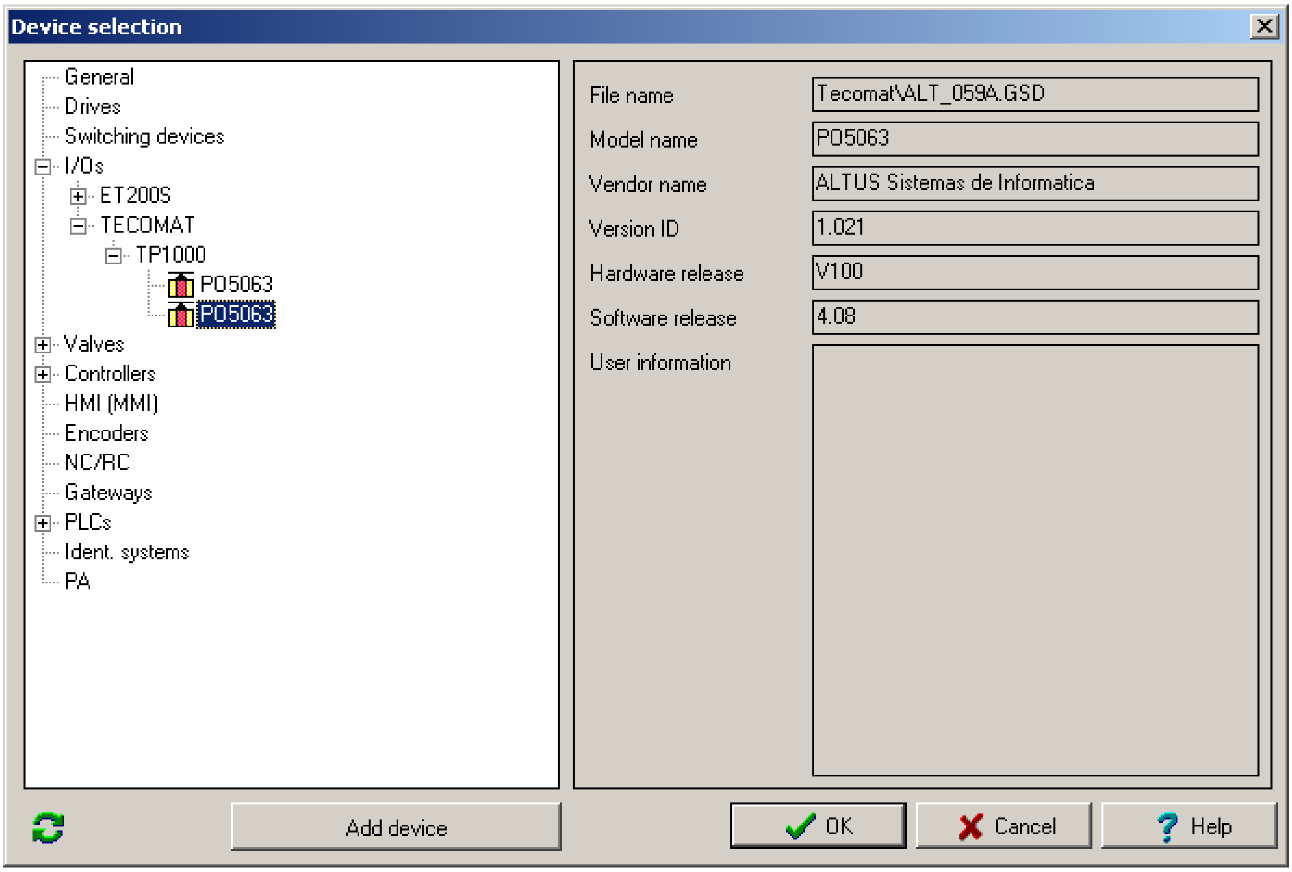
Device selection

From the tree-shaped menu on the left we select the device required. If the device required is not in the menu, information about it must be added via the button *Add device*. Information necessary for operating the slave station in the PROFIBUS DP network is contained in so-called GSD-file which must be supplied by every producer. It is a text file with .gs? extension where the third character specifies the language mutation of the description. The norm defines the following alternatives:

.gsd - (default), usually English
.gsg - German
.gse - English
.gsf - French
.gsi - Italian
.gsp - Portuguese
.gss - Spanish

In terms of these rules further alternations can occur. The Czech version will thus have the extension .gsc.

After pressing a button *Add device* a dialogue window for searching the GSD-file in the computer will appear. After the required version is found using the extension search, the file is automatically copied to the subdirectory *GSD Files* located in the directory where the Mosaic development environment is installed. From this moment the device description is part of the Mosaic development environment. The device description is inserted to the tree menu according to the rules specified by the producer. Individual device categories listed in the tree are given by norm and the producer classifies its products into some of these categories. For example, the system of remote peripheries TECOMAT TP1000 will always be filed into the category I/O where the directory TECOMAT and the subdirectory TP1000 are stored. Information about classification should be given in the manual describing the relevant device.



File name: Tecomat\ALT_059A.GSD
Model name: PO5063
Vendor name: ALTUS Sistemas de Informatica
Version ID: 1.021
Hardware release: V100
Software release: 4.08
User information:

Fig.2.7.3 The selection of the device representing the slave station

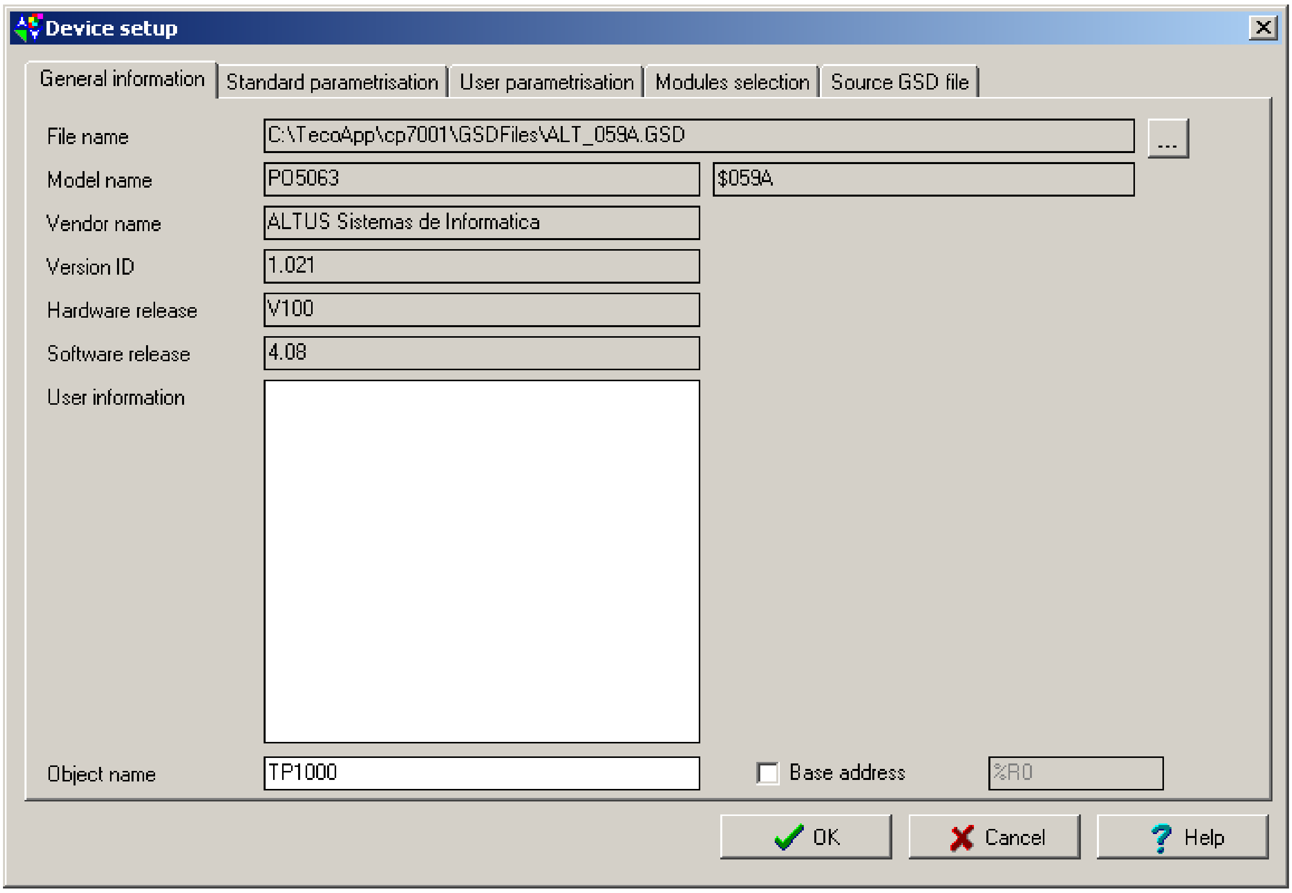
Device setup

After pressing the OK button the panel *Device setup* will appear. Here, we can find the entire configuration of the station necessary for the operation of the slave station in the PROFIBUS DP network. In the tag *General information* there is information about the station and its GSD file can be found (fig. 2.7.4).

Into the space *User information* any text can be entered for easier station identification. If we want to change the GSD file by pressing a push-button ... at the end of the first line the panel *Device selection* will appear (fig. 2.7.3).

In the field *Object name* we enter a name that will become a part of symbolic names of data zones of this station (fig. 2.7.10). The name must be unique within the network (it must not be repeated).

An identification number is important for the PROFIBUS DP network which unequivocally assigns the GSD file to the concrete device. This identification number is displayed on the line *Model name* in the second column (fig. 2.7.4 the number $059A). This number is a part of the GSD file name (5th to 8th name character) and in the case of coincidence with the device used, it appears in the diagnostic zone of the station (see chapter 2.7.3.).



Device setup

File name: C:\TecoApp\cp7001\GSDFiles\ALT_059A.GSD

Model name: PO5063 | $059A

Vendor name: ALTUS Sistemas de Informatica

Version ID: 1.021

Hardware release: V100

Software release: 4.08

User information:

Object name: TP1000

□ Base address: %R0

Obr.2.7.4 Main information on station PROFIBUS DP slave

Standard parameterisation

The tag *Standard parameterisation* allows the setting of standard parameters of the PROFIBUS DP slave station (fig. 2.7.5).

We always enable the option *Lock station for control by this master station* which enables the parameterised slave station to be possible to control from the PLC.

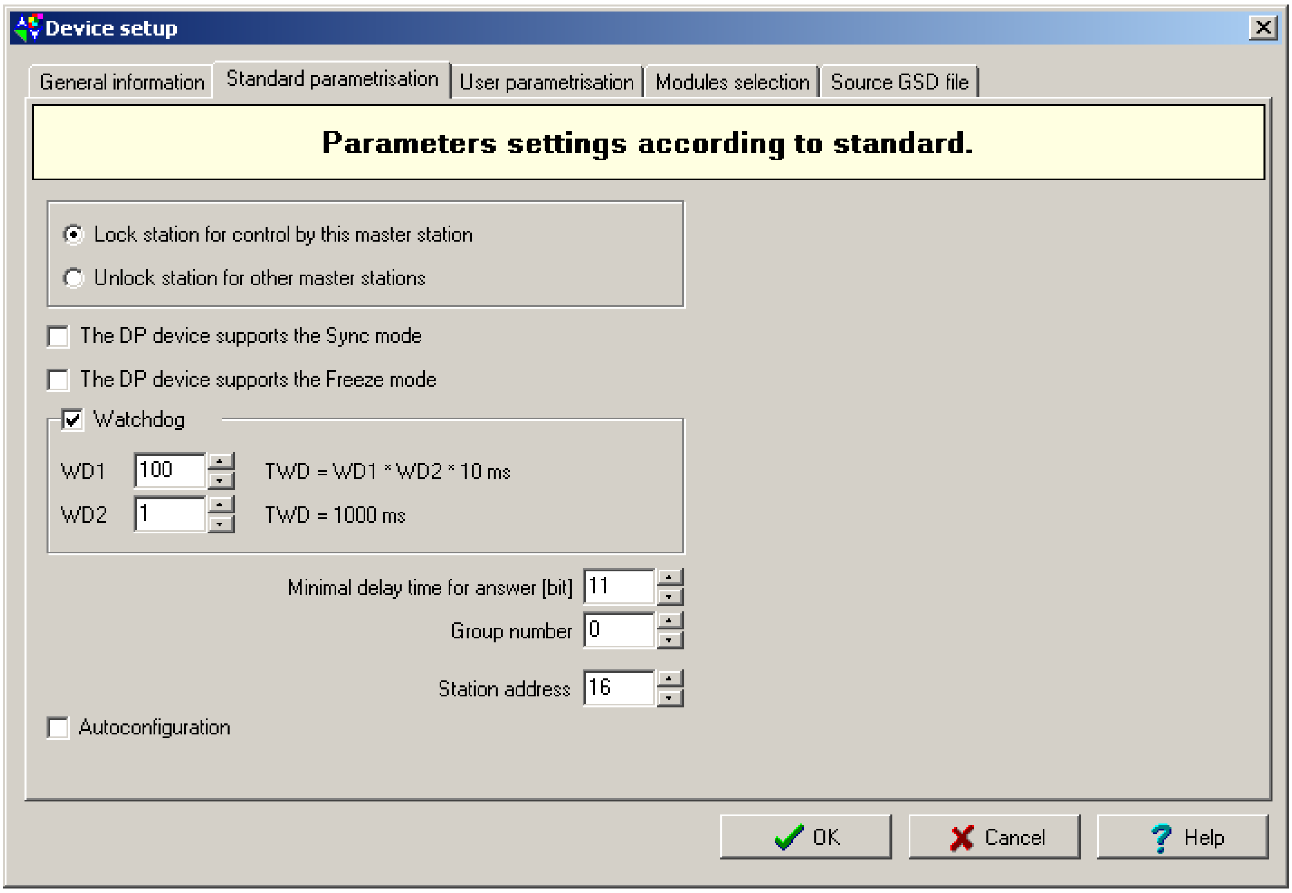
If the station allows, it is possible to enable options *The DP device supports the Sync mode* and *The DP device supports the Freeze mode* (see chapter 2.7.3.6.).

We recommend always to enable option *Watchdog* which is the activation of the check timer of the PROFIBUS DP bus. It ensures that if the station does not receive valid data within a certain time, it switches to a safe status (output blocking, etc.). This time is set by means of two coefficients WD1 and WD2 according to the following equation:

TWD = WD1 * WD2 * 10 ms

It results from the equation that any of these coefficients must not have the 0 value . The resulting value can be seen immediately after the entry of coefficients on the right under the formula in the following form:

TWD = ... ms



Parameters settings according to standard.

● Lock station for control by this master station
○ Unlock station for other master stations

□ The DP device supports the Sync mode
□ The DP device supports the Freeze mode
☑ Watchdog

WD1 100 TWD = WD1 * WD2 * 10 ms
WD2 1 TWD = 1000 ms

Minimal delay time for answer [bit] 11
Group number 0
Station address 16

□ Autoconfiguration

Fig. 2.7.5 Basic parameterisation of the station PROFIBUS DP slave

Minimal delay time for answer is a guaranteed minimum time that elapses after the start of the response transmission. The value is specified through the number of bits and can lay within the range from 11 to 60. Since the TECOMAT PLC has a fast operation of serial channels integrated and it is able to switch to the response reception almost immediately after a query is sent, we set the minimum value of 11.

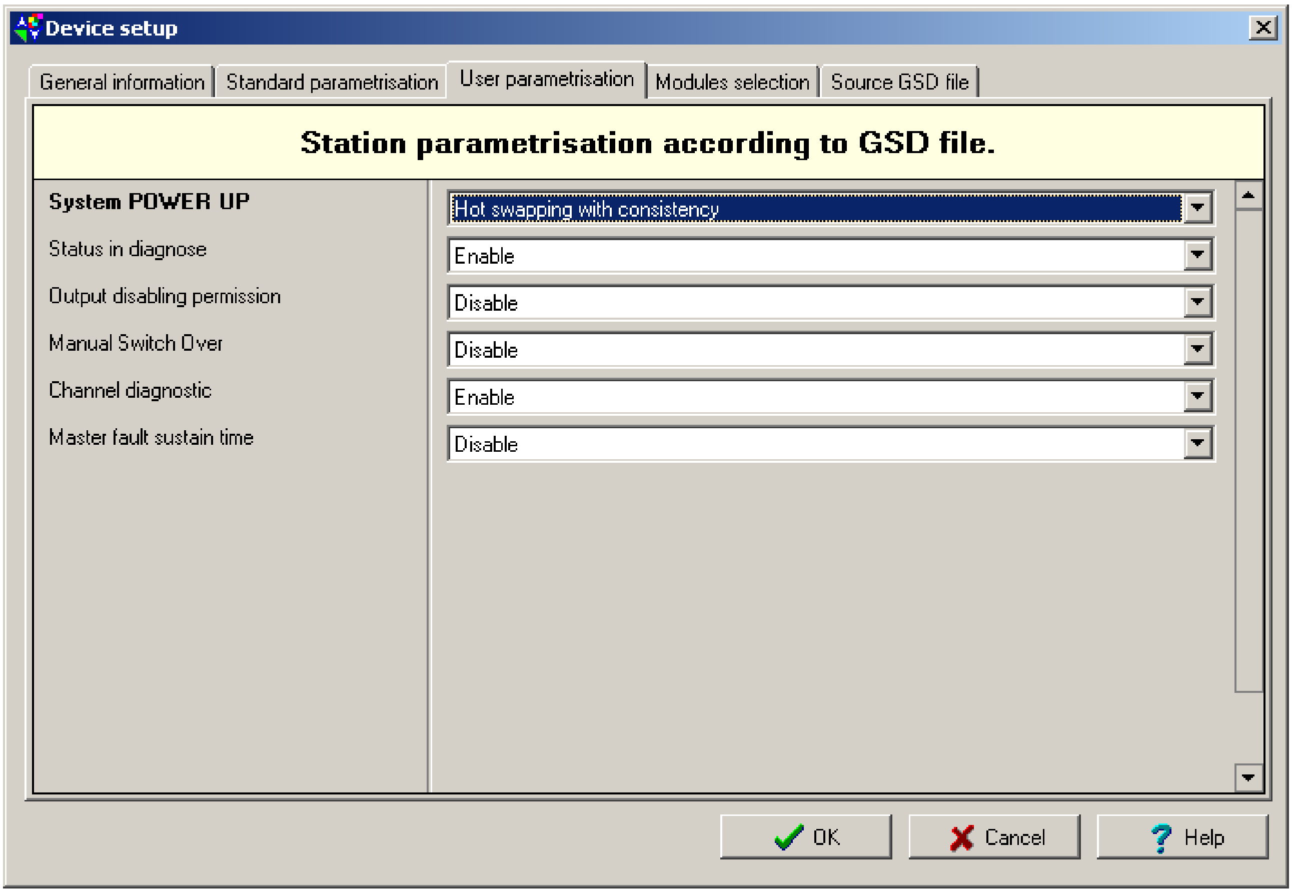
Using a parameter *Group number,* we can create logic groups of stations for the global control of their mode (see chapter 2.7.3.6.). The 0 value represents that the station is not a member of any group.

We also have to set the item *Station address*. Each station of the network including the master station must have its exclusive address within the range from 1 to 125. There must not be two stations with the same address in one network.

If we want to use the station in its basic configuration (a typical example for binary and other single modules or the identification of module configuration when reactivating the system), we can enable the option *Autoconfiguration*. The principle is that such a configuration is used which is offered by the slave station as the initial. The PLC downloads the configuration directly from the slave station (see chapter 2.7.3.4.).

User parameterisation

The tag *User parameterisation* enables the setting of parameters of the PROFIBUS DP slave station defined by the producer (fig. 2.7.6). All these settings are given by the GSD file. It means that also the language used is given be the GSD file.



Station parametrisation according to GSD file.

System POWER UP	Hot swapping with consistency
Status in diagnose	Enable
Output disabling permission	Disable
Manual Switch Over	Disable
Channel diagnostic	Enable
Master fault sustain time	Disable

Fig. 2.7.6 User parameterisation of the station PROFIBUS DP slave (here TP1000 in the English version according to the file ALT_059A.GSD)

Modules selection

The tag *Modules selection* allows the configuration of the PROFIBUS DP slave station (fig. 2.7.7). The list is again given by the GSD file. A detailed explanation, what each module means and what function it has, should be specified by the producer in the documentation.

It is generally valid that the usable modules are listed in the left column and they can be moved to the right column using the button *Add* that represents the current station configuration. The transfer can be undertaken using the mouse, too (Drag and Drop). The module from the right column can be removed by the button *Delete* and by using arrow keys the order of the modules can be changed.

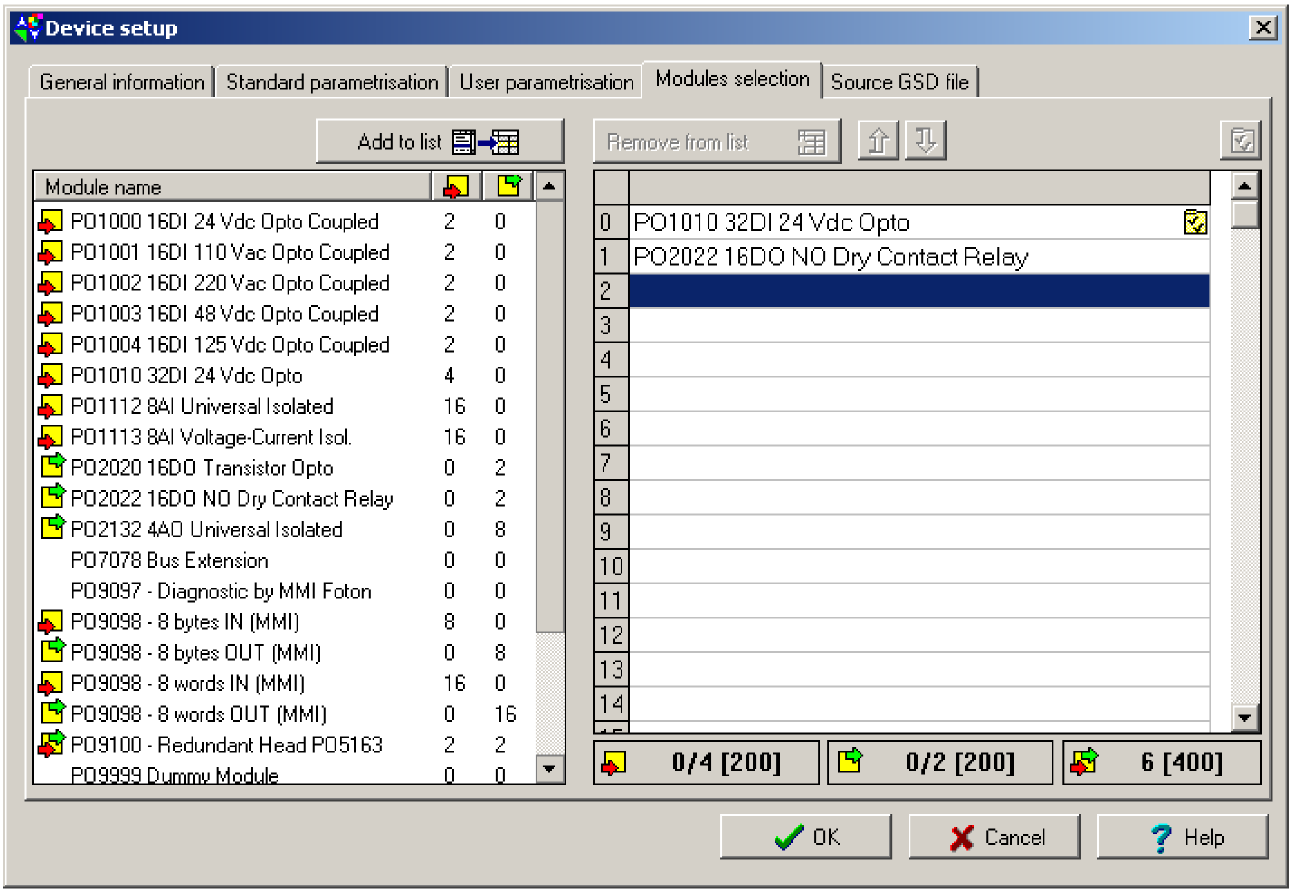
If the module has any parameters, an icon ☑ appears on its line and by pressing the icon a panel with the settings of these parameters will appear (again, it is given by the GSD file).

For each module in the list the number of input and output bytes is specified. At the bottom in the right column, there are three current data with the number of data transmitted. Inputs are on the left, outputs are in the middle and inputs and outputs together are on the right. Data are shown in the following form:

a / b [c]

where a represents the number of input / output data of the selected module in the right column;
b represents the number of input / output data of all modules in the right column;
c represents the maximum number of input / output data that can be transmitted by the station.

Data **a** is not specified for inputs and outputs together.



Module name	⇦	⇨
PO1000 16DI 24 Vdc Opto Coupled	2	0
PO1001 16DI 110 Vac Opto Coupled	2	0
PO1002 16DI 220 Vac Opto Coupled	2	0
PO1003 16DI 48 Vdc Opto Coupled	2	0
PO1004 16DI 125 Vdc Opto Coupled	2	0
PO1010 32DI 24 Vdc Opto	4	0
PO1112 8AI Universal Isolated	16	0
PO1113 8AI Voltage-Current Isol.	16	0
PO2020 16DO Transistor Opto	0	2
PO2022 16DO NO Dry Contact Relay	0	2
PO2132 4AO Universal Isolated	0	8
PO7078 Bus Extension	0	0
PO9097 - Diagnostic by MMI Foton	0	0
PO9098 - 8 bytes IN (MMI)	8	0
PO9098 - 8 bytes OUT (MMI)	0	8
PO9098 - 8 words IN (MMI)	16	0
PO9098 - 8 words OUT (MMI)	0	16
PO9100 - Redundant Head PO5163	2	2
PO9999 Dummy Module	0	0

0	PO1010 32DI 24 Vdc Opto
1	PO2022 16DO NO Dry Contact Relay
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

⇦	⇨	⇔
0/4 [200]	0/2 [200]	6 [400]

Fig. 2.7.7 Configuration of the station PROFIBUS DP slave (here TP1000)

Source GSD file

The tag *Source file GSD* allows the direct viewing of the GSD file content according to which the slave station is set (fig. 2.7.8)..

Important information that can be found in the GSD file is as follows:

```
Ident_Number = 0x059A
```

The item *Ident_Number* specifies the identification number of device.

```
9.6_supp = 1                    ;   9.6  kBaud
19.2_supp = 1                   ;  19.2  kBaud
93.75_supp = 1                  ;  93.75 kBaud
187.5_supp = 1                  ; 187.5  kBaud
500_supp = 1                    ; 500    kBaud
1.5M_supp = 1                   ;   1.5  MBaud
3M_supp = 1                     ;   3    MBaud
6M_supp = 1                     ;   6    MBaud
12M_supp = 1                    ;  12    MBaud
```

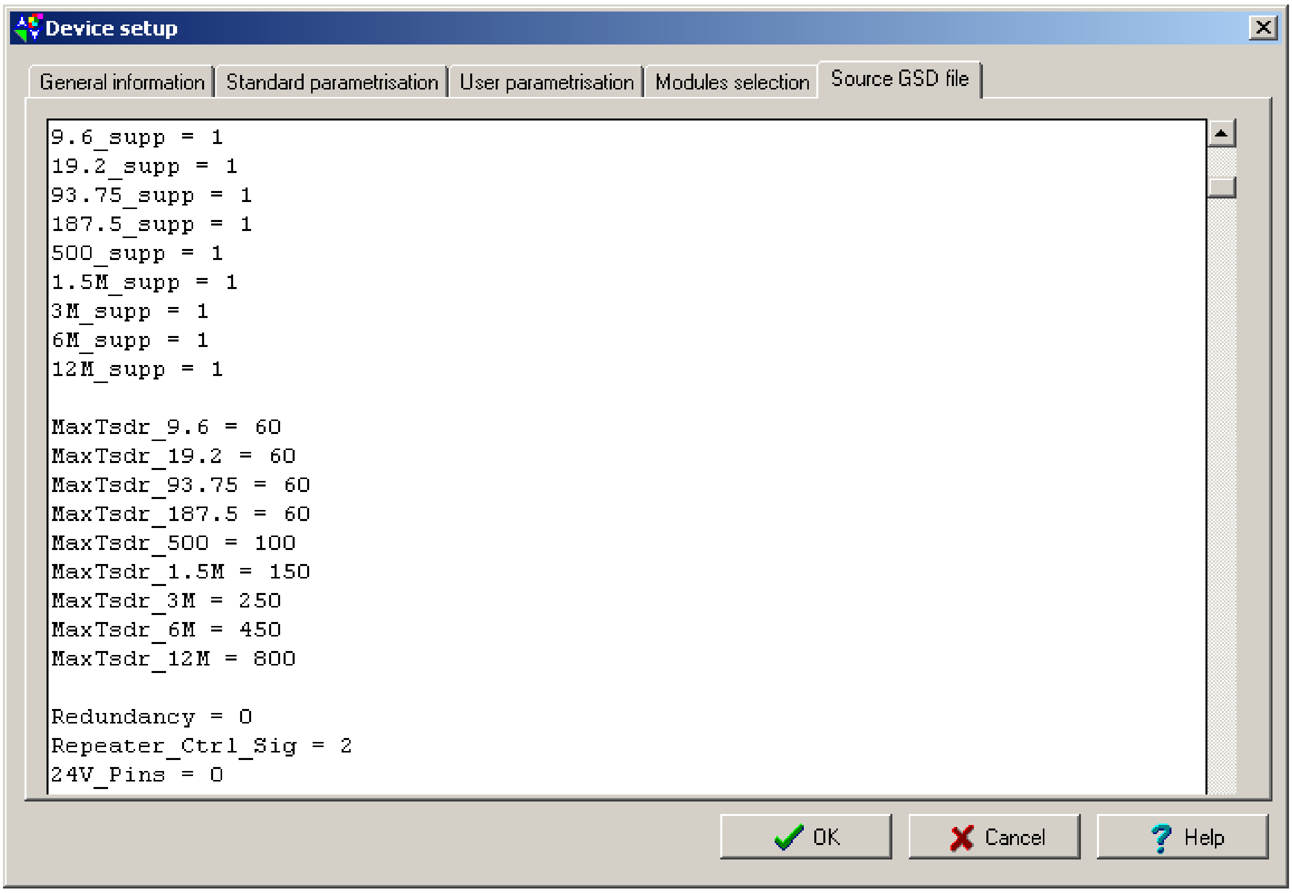
Here is a list of available transmission rates. If we do not find here at least one of items 187.5_supp = 1, 93.75_supp = 1, 19.2_supp = 1, or 9.6_supp = 1, the station cannot be connected since it does not support any of these rates (naturally, all slave stations connected should "agree" on the same rate).

```
Freeze_Mode_supp = 1            ; Freeze-Mode
Sync_Mode_supp = 1              ; Sync-Mode
```

The station supports modes Sync a Freeze.

```
Auto_Baud_supp = 1              ; Autobaud
```

The station sets the transmission rate automatically according to the situation in the network.



```
9.6_supp = 1
19.2_supp = 1
93.75_supp = 1
187.5_supp = 1
500_supp = 1
1.5M_supp = 1
3M_supp = 1
6M_supp = 1
12M_supp = 1

MaxTsdr_9.6 = 60
MaxTsdr_19.2 = 60
MaxTsdr_93.75 = 60
MaxTsdr_187.5 = 60
MaxTsdr_500 = 100
MaxTsdr_1.5M = 150
MaxTsdr_3M = 250
MaxTsdr_6M = 450
MaxTsdr_12M = 800

Redundancy = 0
Repeater_Ctrl_Sig = 2
24V_Pins = 0
```

Fig. 2.7.8 GSD source file of the PROFIBUS DP slave station

The station setup is confirmed by pressing the button *OK* and then we go back to the Project manager. On the desktop, the object with a title PROFIBUS appears with the field where we can enter the name. If we need to modify the settings of any parameter, we use the mouse right click above the selected object and select the option *Network settings*. Again we activate the panel *Device setup*.

Network creation

We gradually add the required number of PROFIBUS DP slave stations to the desktop. Using the option *Objects* | *Bus* we draw the bus. Then we connect all PROFIBUS DP slave stations and the PLC serial channel in the PFB mode to the bus (we always click with the mouse on the object channel and on the bus), this way we create the entire network (fig. 2.7.9).

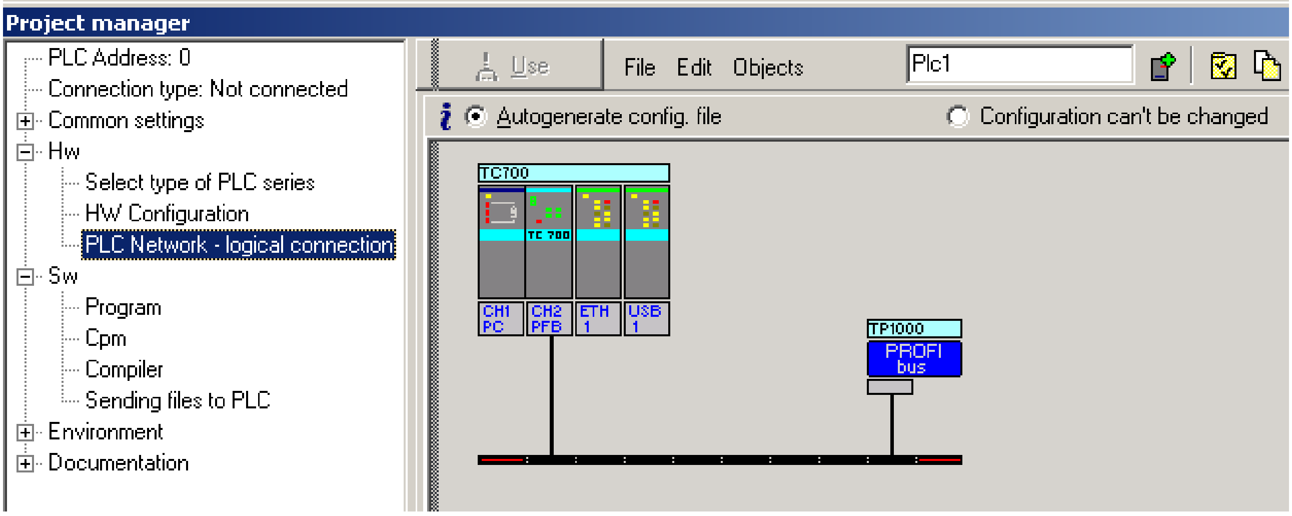


Fig. 2.7.9 PROFIBUS DP network creation

2.7.3. Network operation

Network operation runs on the background of the user program asynchronously to the user program cycle period. The current network data are stored in the internal memory of the system and the data exchange with the scratchpad proceeds always at the I/O scan of the user program.

Operation diagnostics and network data

Serial channels in the **PFB** mode release diagnostics data of network status, data exchanged with another participants and the status of communication with these participants. This data are stored in the scratchpad and are easily accessible from the panel *I/O setting*, accessible via the icon IO (fig. 2.7.10).

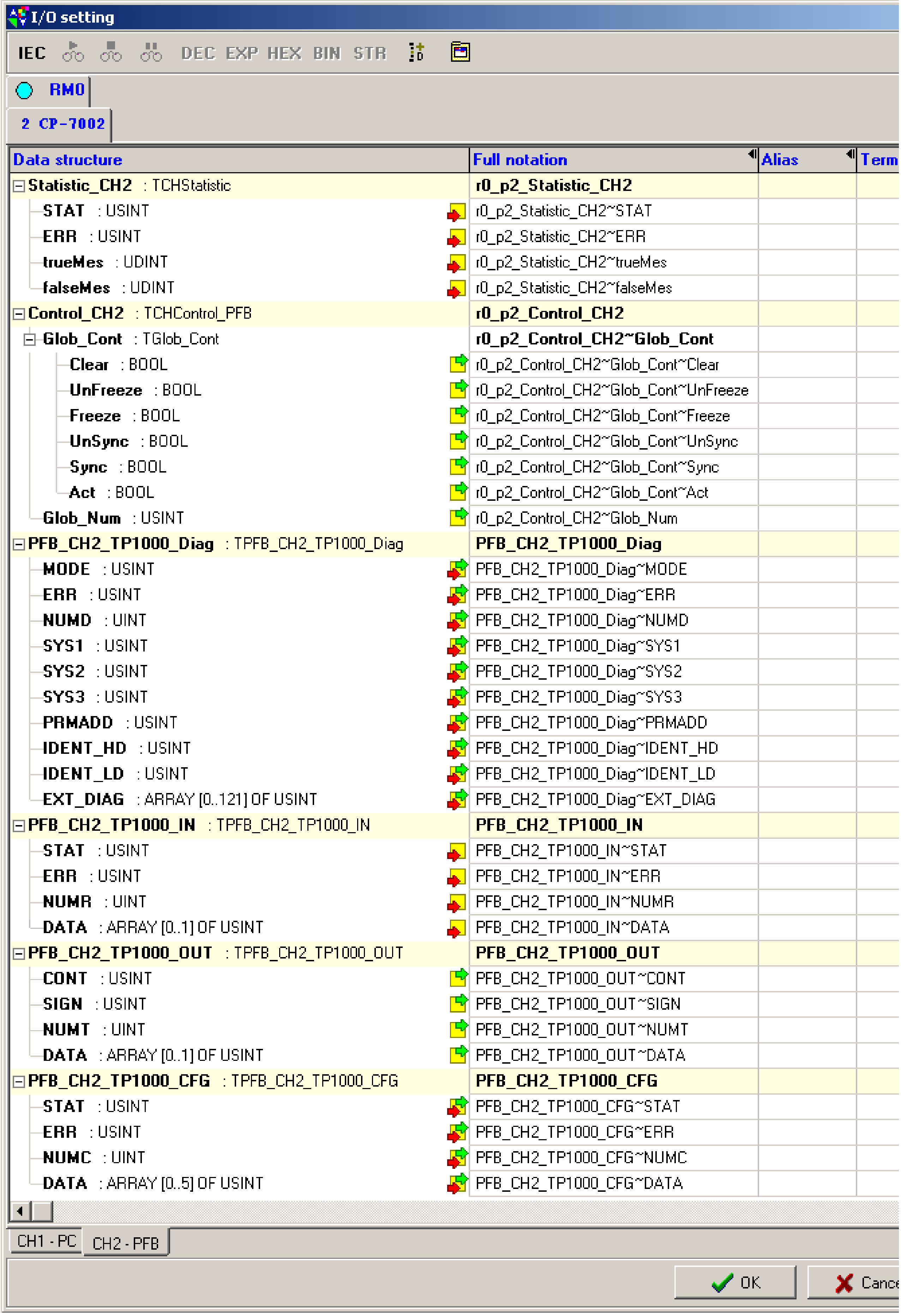
Data have assigned symbolic names that begin with the rack and position number. In the column *Full notation* the concrete symbolic name for the given item is always specified.

If we want to use data in the user program, we use either this symbolic name or we enter our symbolic name which we then can use in the column *Alias*. In no case do we use absolute operands since they can alter after a new compilation of the user program.

The user enters data that are to be written in the corresponding slave stations, in zones with output data. In zones with input data, there are data transmitted from the relevant slave stations. Before they are processed it is necessary to check the COM bit in the relevant status. The structure of input and output data is assigned by the slave station producer.

In zones with diagnostic data, there are the diagnostic data transmitted from the relevant slave stations. The structure of the data is determined both by the standard and by the slave station producer.

In zones with configuration data, there are current configuration data used for configuration of the relevant slave stations. In case that the auto-configuration is on, it is possible to check what configuration data are used.



Data structure	Full notation	Alias	Term
⊟Statistic_CH2 : TCHStatistic	r0_p2_Statistic_CH2		
STAT : USINT	r0_p2_Statistic_CH2~STAT		
ERR : USINT	r0_p2_Statistic_CH2~ERR		
trueMes : UDINT	r0_p2_Statistic_CH2~trueMes		
falseMes : UDINT	r0_p2_Statistic_CH2~falseMes		
⊟Control_CH2 : TCHControl_PFB	r0_p2_Control_CH2		
⊟Glob_Cont : TGlob_Cont	r0_p2_Control_CH2~Glob_Cont		
Clear : BOOL	r0_p2_Control_CH2~Glob_Cont~Clear		
UnFreeze : BOOL	r0_p2_Control_CH2~Glob_Cont~UnFreeze		
Freeze : BOOL	r0_p2_Control_CH2~Glob_Cont~Freeze		
UnSync : BOOL	r0_p2_Control_CH2~Glob_Cont~UnSync		
Sync : BOOL	r0_p2_Control_CH2~Glob_Cont~Sync		
Act : BOOL	r0_p2_Control_CH2~Glob_Cont~Act		
Glob_Num : USINT	r0_p2_Control_CH2~Glob_Num		
⊟PFB_CH2_TP1000_Diag : TPFB_CH2_TP1000_Diag	PFB_CH2_TP1000_Diag		
MODE : USINT	PFB_CH2_TP1000_Diag~MODE		
ERR : USINT	PFB_CH2_TP1000_Diag~ERR		
NUMD : UINT	PFB_CH2_TP1000_Diag~NUMD		
SYS1 : USINT	PFB_CH2_TP1000_Diag~SYS1		
SYS2 : USINT	PFB_CH2_TP1000_Diag~SYS2		
SYS3 : USINT	PFB_CH2_TP1000_Diag~SYS3		
PRMADD : USINT	PFB_CH2_TP1000_Diag~PRMADD		
IDENT_HD : USINT	PFB_CH2_TP1000_Diag~IDENT_HD		
IDENT_LD : USINT	PFB_CH2_TP1000_Diag~IDENT_LD		
EXT_DIAG : ARRAY [0..121] OF USINT	PFB_CH2_TP1000_Diag~EXT_DIAG		
⊟PFB_CH2_TP1000_IN : TPFB_CH2_TP1000_IN	PFB_CH2_TP1000_IN		
STAT : USINT	PFB_CH2_TP1000_IN~STAT		
ERR : USINT	PFB_CH2_TP1000_IN~ERR		
NUMR : UINT	PFB_CH2_TP1000_IN~NUMR		
DATA : ARRAY [0..1] OF USINT	PFB_CH2_TP1000_IN~DATA		
⊟PFB_CH2_TP1000_OUT : TPFB_CH2_TP1000_OUT	PFB_CH2_TP1000_OUT		
CONT : USINT	PFB_CH2_TP1000_OUT~CONT		
SIGN : USINT	PFB_CH2_TP1000_OUT~SIGN		
NUMT : UINT	PFB_CH2_TP1000_OUT~NUMT		
DATA : ARRAY [0..1] OF USINT	PFB_CH2_TP1000_OUT~DATA		
⊟PFB_CH2_TP1000_CFG : TPFB_CH2_TP1000_CFG	PFB_CH2_TP1000_CFG		
STAT : USINT	PFB_CH2_TP1000_CFG~STAT		
ERR : USINT	PFB_CH2_TP1000_CFG~ERR		
NUMC : UINT	PFB_CH2_TP1000_CFG~NUMC		
DATA : ARRAY [0..5] OF USINT	PFB_CH2_TP1000_CFG~DATA		

*Fig. 2.7.10 Serial channel data in the **PFB** mode*

Network diagnostics:
Input data *rx_py_Statistic_XXX* (*TCHStatistic* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Stat - communication status (8-times bool type)

	X	X	X	X	X	X	CLE	CLI
bit	7	6	5	4	3	2	1	0

CLE - 1 - master is at status Clear due to external requirement (the user set the requirement via the *Glob_Cont variable*)
CLI - 1 - master is at status Clear due to internal requirement (the entry initialization of slave stations runs

Err - communication error (see chapter 2.7.4) (usint type)
trueMes - number of valid network communication (udint type)
falseMes - number of invalid network communication (see chapter 2.7.3.2) (udint type)

Output data *rx_py_Control_XXX* (*TCHControl_PFB* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Glob_Cont network mode global control (usint type)
Possibility of the network control by synchronisation commands (see chapter 2.7.3.6.).

Glob_Num group number (usint type)
Number of the logic group of slave stations which the synchronisation command is destined for.

Diagnostic data *PFB_XXX_NNN_Diag* (*TPFB_XXX_NNN_Diag* structure):

(XXX - communication channel designation, NNN - object name set according to fig. 2.7.4)

Mode - mode of communication with the slave station (usint type)
The variable takes values 0 to 9 according to the status diagram in fig. 2.7.1.

Err - communication error with the station (see chapter 2.7.4) (usint type)

NumD - diagnostic data length from the slave station (uint type)

Sys1 system diagnostics 1 (8-times bool type)

	MLO	PRF	ISR	NSP	EXD	CFF	SNR	NCO
bit	7	6	5	4	3	2	1	0

MLO - 1 - slave station is parameterised by another master station, parameterisation is not possible
PRF - 1 - invalid parameterisation (e.g. wrong identification number, etc.)
ISR - 1 - invalid response of the slave station (connection lost or interfered, input and output data is not refreshed)
NSP - 1 - slave station does not support the station function (the station is in a status when it is not possible to exchange data)

EXD - 1 - slave station sends additional diagnostic data Ext_Diag given by the producer
CFF - 1 - configuration data does not match
SNR - 1 - slave station is not ready for the data exchange (the station is in the phase of parameterisation or configuration expectation)
NCO - 1 - slave station not found (initial status before communication establishing)

Sys2 system diagnostics 2 (8-times bool type)

	DGD	0	SNC	FRZ	WDO	1	STD	PRR
bit	7	6	5	4	3	2	1	0

DGD - 1 - diagnostics data not valid (connection lost)
SNC - 1 - command Sync received (see chapter 2.7.3.6.)
FRZ - 1 - Freeze command received (see chapter 2.7.3.6.)
WDO - 1 - active check timer of the watchdog bus
STD - 1 - static diagnostics - slave station cannot temporarily pass data due to application failure (e.g. I/O-circuits power supply failure, application program stopped)
PRR - 1 - request for new parameterisation and configuration (serial channel reacts automatically by repeating parameterisation and configuration)

Sys3 system diagnostics 3 (8-times bool type)

	EXO	0	0	0	0	0	0	0
bit	7	6	5	4	3	2	1	0

EXO 1 - additional diagnostic data overflow Ext_Diag

Prm_Add address of parameterising master station (usint type)
Master station address that performed parameterisation of the slave station. Here, the value must appear which we specified as the address of the serial channel in the **PFB** mode.

Ident_HD
Ident_LD station identification number (2-times usint type)
The identification number is fix-determined for each type of PROFIBUS DP slave station. This number must correspond with the identification number in the GSD-file.

Ext_Diag [x] additional diagnostics data (usint array element type)
Further diagnostics data typical for the given type of the slave station. The description of this data must be received from the slave station producer.

Data exchanged with slave station:
Input data *PFB_XXX_NNN_IN* (*TPFB_XXX_NNN_IN* structure):

(XXX - communication channel designation, NNN - object name set according to fig. 2.7.4)

Stat - status of communication with the slave station (8-times bool type)

	X	X	X	X	X	X	COM	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

COM - data exchange status
0 - data exchange not running, the following data is not valid
1 - data exchange running, the following data is valid

Err - communication error with the station slave (see chapter 2.7.4) (usint type)
NumR - length of read data from the slave station (uint type)
Data [x] - data read from the slave station (usint array element type)

Output data *PFB_XXX_NNN_OUT* (*TPFB_XXX_NNN_ OUT* structure):

(XXX - communication channel designation, NNN - object name set according to fig. 2.7.4)

Cont - communication control (8-times bool type)

	RST	X	X	X	X	X	X	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

RST - communication reset with the slave station
0 - leave the communication mode without a change
1 - mode of communication with the slave station (see variable *Mode* in diagnostics data) is set to 0 and so new communication establishment with the slave station is reactivated, as well as new parameterisation and configuration.

Sign - reserve (usint type)
NumT - data length entered to the slave station (uint type)
Data [x] - data entered to the slave station (usint array element type)

Configuration data *PFB_XXX_NNN_CFG* (*TPFB_XXX_NNN_CFG* structure):

(XXX - communication channel designation, NNN - object name set according to fig. 2.7.4)

Stat - slave station configuration status (8-times bool type)

	X	X	X	X	X	X	COM	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

COM - validity of configuration data
0 - configuration data is not valid
1 - configuration data is valid

Err - communication error with the participant (see chapter 2.7.4) (usint type)
NumC - length of configuration data for the slave station (uint type)
Data [x] -current configuration data for the slave station (see chapter 2.7.3.4.) (usint array element type)

2.7.3.1. Network start

Before the connection of slave stations to the PROFIBUS network, it is necessary to set transmission parameters. Most of these stations have automatic transmission rate detection so it does not need to be set. Attention should be paid to the setting of addresses of individual stations. Each station in the network including the master station must have its exclusive address within the range of 1 to 125. Two stations with the same address must not be in one network. The master station address and the transmission rate in the network are part of the settings of the serial channel (see chapter 2.7.1.).

2.7.3.2. Communication establishment

After the start of the user program, PLC starts to establish communication with slave stations specified in the network initialisation (see chapter 2.7.2.). If communication with the slave station is established, the PLC starts its parameterisation. If not, it will keep trying to establish the communication.

Note: Slave stations that have the automatic detection of the transmission rate, first, monitor the operation on the line and according to it they set their own rate. Due to this reason, there nearly always appear several error communications during communication establishment within the *falseMes* variable in the Network Diagnostics. This status does not indicate the error. It is important that error communications will not accumulate during the data exchange.

2.7.3.3. Station parameterisation

After a successful communication establishment, the PLC performs so-called slave station parameterisation which serves for setting of all parameters of the slave station. These parameters are always contained in tags *Standard parameterisation* and *User parameterisation* that belongs to the panel *Device setup* (see chapter 2.7.2.).

If the parameterisation is performed successfully, the PLC starts the slave station configuration. If not, it will keep trying to perform parameterisation. If the connection is lost, the PLC switches back to the state of establishing communication again.

2.7.3.4. Station configuration

After a successful parameterisation, the PLC performs the slave station configuration. The configuration is used for setup of the volume of transmitted data from / to the slave station. This data is generated by the Mosaic development environment from the tag *Modules selection* within the panel *Device setup* (see chapter 2.7.2.). The current configuration is entered to the configuration zone in the PLC scratchpad.

If the configuration is entered successfully, the lengths of transmitted data calculated from configuration values appear in data zones. The PLC starts to exchange data with the slave station. If the configuration is not performed successfully, communication is switched back to the parameterisation. If the connection is lost, the communication switches back to the phase of communication establishment.

Auto-configuration

If we want to use the station with its initial configuration (a typical example is the connection of another PLC with variable data length or detection of station configuration when reactivating the network), we can use so-called auto-configuration (enabling the option *Autoconfiguration* in the tag *Standard parameterisation* within the panel *Device setup*). The principle is, that such a configuration is used which is offered by the slave station as the initial. The PLC downloads the configuration directly from the slave station.

Attention! Some stations have the initial configuration empty, and it is changed only by entering the configuration from the master station. For these stations, auto-configuration does not work. It usually concerns modular remote I/O or DP/DP couplers.

On the other hand, this application can be advantageous in connection with another PLC or generally with such a device which sets its initial configuration according to its own current programming. The number of transmitted bytes is automatically adjusted on the side of the master station according to the current status of the slave station.

The behaviour of the slave station during auto-configuration must be examined first!

Downloaded configuration data are entered to the configuration zone. The structure of configuration data for one data module is as follows:

CfgBn configuration byte n
- general format

	Con	Size	Output	Input	Length			
bit	7	6	5	4	3	2	1	0

Con data time consistency
0 - consistency through byte / word
1 - consistency through total data length
Size data format
0 - byte
1 - word
Output 1 - output data
Input 1 - input data
Length data number – 1
0 - 1 byte / word
1 - 2 byte / word
:
15 - 16 byte / word

- special format

	Next		0	0	Prod			
bit	7	6	5	4	3	2	1	0

Next next information
00 - none
01 - length byte for input data follows
10 - length byte for output data follows
11 - follows after one length byte for input and output data
Prod length of data specified by the producer (0 - 14)

LenBn length byte n (for special format only)

	Con	Size	Length					
bit	7	6	5	4	3	2	1	0

Con data time consistency
0 - consistency through byte / word
1 - consistency through total data length
Size data format
0 - byte
1 - word
Length number of data – 1
0 - 1 byte / word
1 - 2 byte / word

:
63 - 64 byte / word

ProdB byte n specified by the producer (for special format only)

Configuration can be specified in two formats - general and special. For example, 4 input bytes and 4 output bytes in the general format have the following form:

$13, $23

In the special format, the same configuration may look as follows:

$40, $03, $80, $03

The possibilities of the slave station configuration are determined by the station producer. Information can be found also in the GSD-file supplied by the producer.

2.7.3.5. Data exchange

After a successful configuration cyclic data exchange can be performed without a problem. For this purpose, data zones in the PLC scratchpad memory are used. The zone contains communication status, length of transmitted data calculated by the PLC based on configuration data, other bytes contain the data itself (see chapter 2.7.3.).

In the input zone, there are input data that are read from the slave station *n* and on the other hand, in the output zone the PLC expects output data which are then entered to the slave station *n*. If the slave station does not have any input data, the value of the data length in the input zone is 0. If the slave station does not require output data, the value of the data length in the output zone is 0. However, both zones exist and the status and control variables are functional. The structure of the input and output data is given by the producer.

Conversion type

The PROFIBUS DP bus supports data transmission of byte and word width where every data module of the slave station represents either a byte type data array or word byte data array (see chapter 2.7.3.4.). The **PFB** mode ensures bidirectional conversion of the word data type in such a way that in the TECOMAT PLC scratchpad the rule is applied that the lower byte has a lower address (Intel convention) while on the PROFIBUS DP bus it is reversely (Motorola convention).

Attention should be paid to data storage at each slave station being connected. If the data of 8 bits width are transmitted as the byte type data array and the data of 16 bits width are transmitted as the word type data array, then everything is OK. Both array types can be combined if the slave station is modular. If the transmitted data have more complex structure and are transmitted by the PROFIBUS DP bus as byte arrays then for all data we must swap bytes larger than 8 bits width using instructions SWP and SWL in the user program. If we do not do this, we can access correctly only the individual data bytes.

For example, data transmitted have the following structure:

	D16		D8	D8	D32			
byte	0	1	2	3	4	5	6	7

Data are transmitted by the PROFIBUS DP bus in the following order:

	D16H	D16L	D8	D8	D32D	D32C	D32B	D32A
byte	0	1	2	3	4	5	6	7

If data are transmitted as the byte type array, then these data are saved in the PLC scratchpad. Then, however, only the individual bytes of data D16 and D32 can be accessed. Using instructions SWP and SWL we must modify data into the following form:

	D16L	D16H	D8	D8	D32A	D32B	D32C	D32D
byte	0	1	2	3	4	5	6	7

For the conversion of data of 16 bits width we will use the following procedure:

```
LD    D16
SWP
WR    D16
```

For the conversion of data of 32 bits width we will use the following procedure:

```
LD    D32
SWP
SWL
SWP
WR    D32
```

If data are transmitted as the word type array, the situation is different. For example, transmitted data have the following structure:

	D16		D16		D32			
byte	0	1	2	3	4	5	6	7

These data are passed on to the PROFIBUS DP bus in the following order:

	D16H	D16L	D16H	D16L	D32D	D32C	D32B	D32A
byte	0	1	2	3	4	5	6	7

Data transmitted as the word type array are automatically converted to the word type according to PLC TECOMAT and will be stored in the PLC scratchpad as follows:

	D16L	D16H	D16L	D16H	D32C	D32D	D32A	D32B
byte	0	1	2	3	4	5	6	7

If data of 32 bits width are transmitted in this way, the following procedure will be used for their conversion:

```
LD    D32
SWL
WR    D32
```

The same principles for both received and sent data are applied.

2.7.3.6. Global control

By means of this function, it is possible to send commands (Sync, Unsync, Freeze, Unfreeze and Clear Data) to all slave stations or their selected logic groups for the purpose of synchronisation. The classification of the slave station to the logic group is given by item *Group number* in the tag *Standard parameterisation* within the panel *Device setup* (see chapter 2.7.2.). This function is controlled by means of two output bytes of the diagnostics zone of the network (see chapter 2.7.3.):

Glob_Cont network mode global control (usint type)

	Act	0	Sync	UnSync	Freeze	UnFreeze	Clear	Stop
bit	7	6	5	4	3	2	1	0

Act 1 - send command to the PROFIBUS DP network
For sending of any of the below described commands, this bit must be set to log.1. The PLC sets it to zero automatically.
Sync command Sync for output synchronisation
UnSync command UnSync for output unsynchronisation (realising)
Freeze command Freeze for freezing of inputs
UnFreeze command UnFreeze for unfreezing of inputs
Clear command Clear for safe status of outputs
Stop command Stop for network operation termination

Glob_Num group number (usint type)
The number of the logic group of the slave station for which the synchronisation command applies. 0 value is valid for all slave stations in the network.

Commands Sync, UnSync

Commands Sync and UnSync are used for time control of the start-up of instruments belonging to one group. By sending the Sync command (log.1 to *Sync* and *Act* bits) the slave station holds the outputs on the current values. Subsequently sent data remain in the memory of the slave stations only and they will appear on the physical outputs at the moment when the UnSync command is sent (log.1 to *UnSync* and *Act* bits). Since these commands are sent through the global message, they are received by all slave stations at the same time. Therefore, outputs are set at the same time.

Commands Freeze, UnFreeze

Commands Freeze and UnFreeze serve to get the exact image of the process input of one group. By sending the Freeze command (log.1 to *Freeze* and *Act* bits) slave stations scan the inputs and freeze them. Data received contain states of the frozen inputs until the UnFreeze command is sent (log.1 to *UnFreeze* and *Act* bits).

Command Clear

Command Clear (log.1 to *Clear* and *Act* bits) causes switching of slave station outputs to a safe status. The definition of the safe status depends on the type and setting of the slave station (e.g. for binary outputs, it is usually log.0 or zero voltage, as the case may be). The Clear status is terminated by entering the log.0 to bit *Clear* and log.1 to bit *Act*.

During the Clear state period involved by this command, the bit *CLE* in a variable *Stat* in the network diagnostics is set to log.1.

Command Stop

Command Stop (log.1 to *Stop* and *Act* bits) causes the termination of communication with all slave stations. There is no operation on the network and slave stations diverge to the safe status if they have a control timer Watchdog activated.

To cease the Stop status we enter log.0 to *Stop* bit and log.1 to *Act* bit. The network operation will be restored and all slave stations will be parameterised and configured again. (see chapter 2.7.3.2.,chapter 2.7.3.3., chapter 2.7.3.4.).

The command Stop does not take into consideration the number of the group. There is no operation on the network while this command is active.

2.7.4. Error messages

The code of communication error is saved in the variable *Err* of the receiving zone of each slave station.

$01 invalid mode of communication

The value of communication status *Mode* in the diagnostics zone of the station is out of permissible range. The PLC performs the initial setting of the variable *Mode* to 0 which results to communication establishment, parameterisation and configuration of the relevant station.

$08 communication timeout

The slave station did not respond within specified period of time. It is necessary to check the physical connection of the station and its parameters settings (address and transmission rate).

$0F received data length error

A different number of bytes was received than it was expected. The reason can be communication interference or assignment of the same address to more slave stations.

$10 error SD
$11 error LE
$12 error LEr
$13 error SDr
$14 error DA
$15 error SA
$16 error FC
$17 error SSAP
$18 error DSAP
$1E error CHS
$1F error ED

Communication protocol error. The reason can be communication interference or assignment of the same address to more slave stations.

$30 Connect response error
$31 Data_Exch response error
$3B Get_Cfg response error
$3C Diag_Data response error
$3D Set_Prm response error
$3E Chk_Cfg response error

Another response was received than expected. The reason for this can be communication interference or assignment of the same address to more slave stations.

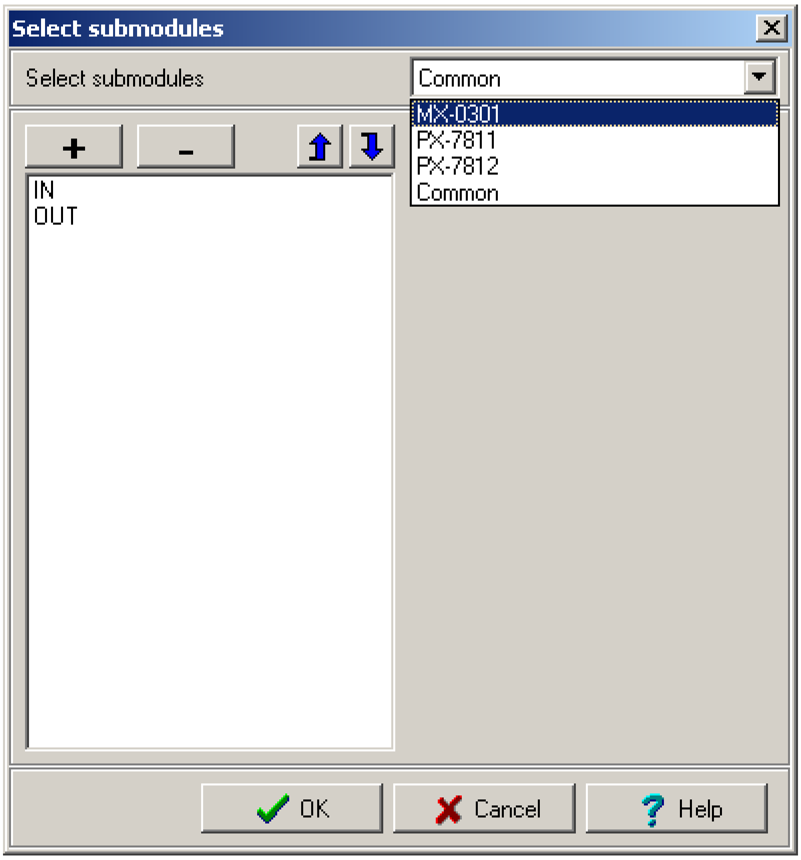
2.8. UPD MODE - OPERATION OF SPECIAL SUBMODULES

The **UPD** mode is used for operation of special submodules that are not only serial channel converters but they have a certain function integrated.

2.8.1. Channel parameters setup

In the Mosaic development environment, we select folder *Hw* | *HW Configuration*. In the list of modules we highlight the central unit, the channel of which we want to set. Then we press the push-button *Settings* or icon ☑ on the line on the right from the position number. Panel *Channel parameters setting* is displayed (fig. 1.1).

We set the corresponding serial channel to **UPD** mode. Then we press button ☑ on this line and window *Select submodules* appears (fig. 2.8.1).



Select submodules

Select submodules: Common

Fig.2.8.1 ***UPD*** *mode parameters setting*

The window has a structure according to a submodule that we select. A detailed setting is described in the documentation supplied with the relevant submodule.

Option *Common* is intended for use of submodules that are not supported so far and with which a regular user does not usually come into contact.

2.9. DPS MODE - PROFIBUS DP SLAVE STATION REALIZATION

What is PROFIBUS DP

PROFIBUS is a data bus used for data transmission in the industrial environment among control rooms, control systems, etc. From the PROFIBUS FMS bus definition the PROFIBUS DP bus was created, used primarily for the connection of remote peripheries (remote I/O), intelligent sensors and actuators. The PROFIBUS DP bus is defined by the IEC 61158 standard and it is supported by a number of manufacturers of various peripheral modules usually having a high class of protection and thus used directly in the technology.

The RS-485 interface used allows a long-distance transmission with a sufficient resistance against interference. Data transmission is performed on master station query → slave station response basis. The message protocol has a multiple data protection ensuring that data successfully transmitted are correct.

DPS mode function

For full connection of PLC TECOMAT to the PROFIBUS DP network, the submodule MR-0152 is used which represents the PROFIBUS DP slave station that is able to communicate within a speed of up to 12 MBd. The data exchange between the submodule and the PLC runs in the **DPS** mode. The submodule MR-0152 contains an ASIC circuit and is fitted with a galvanically isolated RS-485 interface.

The **DPS** mode can be set only when the MR-0152 submodule is fitted in, otherwise, the channel is automatically switched off and the channel initialisation error occurs. Similarly, with the MR-0152 submodule fitted in only the **DPS** or **OFF** mode can be set. Fitting in of modules can be found in the Mosaic development environment in the option *PLC* | *HW configuration* in the tag *Module info* (see chapter 1.1., fig. 1.1).

Attention: **Submodule MR-0152 requires in case of usage within central units CP-7001, CP-7002, CP-7003 TC700 the version hw 02 (for information see fig. 1.1). It does not work with version hw 01.**

2.9.1. Parameters setup

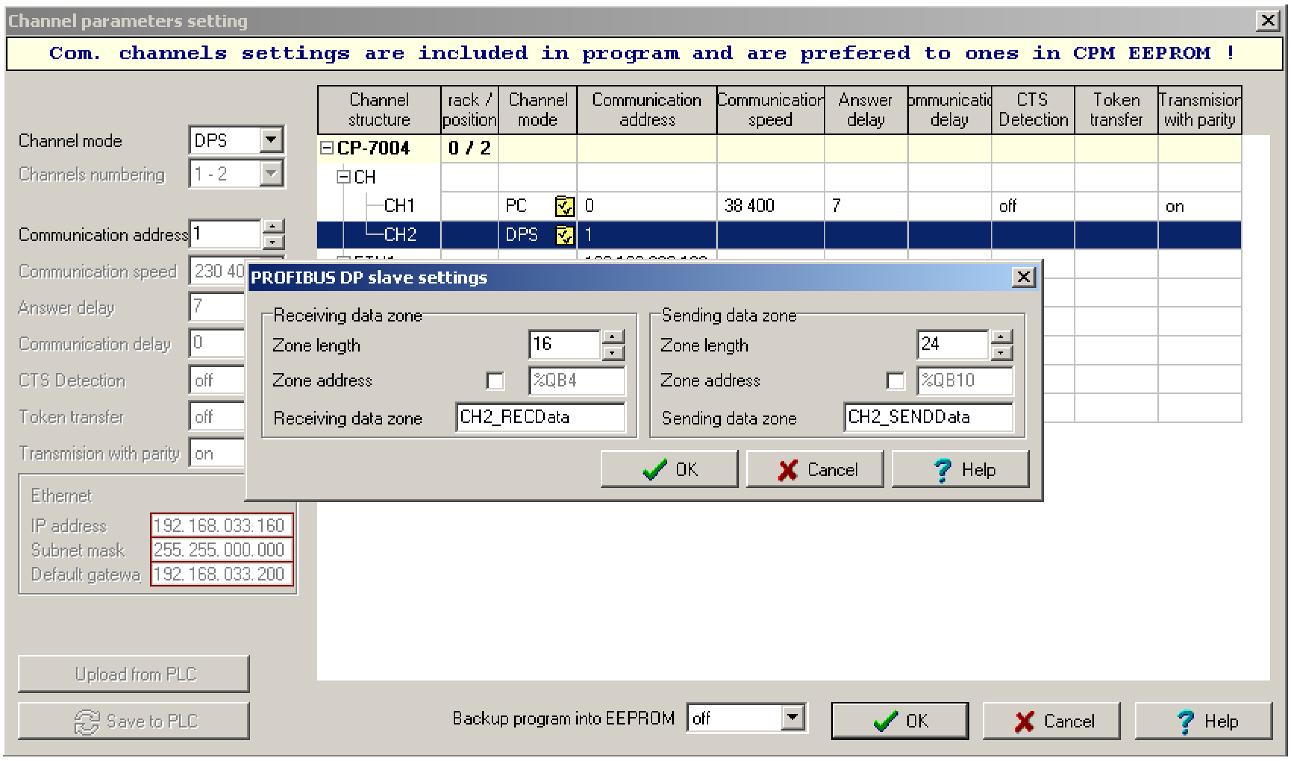
In the Mosaic development environment we select folder *Hw* | *HW Configuration*. In the list of individual modules we highlight the central unit, the channel of which we want to set. Then we press the push-button *Settings* or the icon ☑ on the line on the right from the position number. Panel *Channel parameters setting* is displayed (fig. 1.1).

We set the corresponding serial channel to the **DPS** mode and then we set the address. The address will be interpreted as the slave station address. It must not be identical with any address of any participant of the PROFIBUS DP network and it must not be 0 (address reserved for master of class 2). The transmission rate is not set. The MR-0152 submodule adopts itself automatically to the transmission rate of the PROFIBUS DP master station.

Then we press the button ☑ on this line and a window *PROFIBUS DP slave setting* appears (fig. 2.9.1).

In the **DPS** mode it is necessary to set only the position and the length of received data (output data sent by the master station) and the position and the length of sent data (input data received by the master station). The maximum length of the received or sent data is 240 bytes. The symbolic name (or the physical address rather) indicates received or sent data itself, preceded by four status or four control bytes, as the case may be (see data structure in chapter 2.9.2.).

Since the communication itself is performed by the MR-0152 submodule independently, the central unit processor load is minimal and depends only on the volume of the transmitted data.



Channel parameters setting

Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

Channel mode: DPS
Channels numbering: 1 - 2
Communication address: 1
Communication speed: 230 40
Answer delay: 7
Communication delay: 0
CTS Detection: off
Token transfer: off
Transmision with parity: on

Ethernet
IP address: 192. 168. 033. 160
Subnet mask: 255. 255. 000. 000
Default gatewa: 192. 168. 033. 200

Channel structure	rack / position	Channel mode	Communication address	Communication speed	Answer delay	Communication delay	CTS Detection	Token transfer	Transmision with parity
⊟ CP-7004	0 / 2								
⊟ CH									
CH1		PC	0	38 400	7		off		on
CH2		DPS	1						

PROFIBUS DP slave settings

Receiving data zone
Zone length: 16
Zone address: □ %QB4
Receiving data zone: CH2_RECData

Sending data zone
Zone length: 24
Zone address: □ %QB10
Sending data zone: CH2_SENDData

Backup program into EEPROM: off

Obr.2.9.1 Parameters setting inv the ***DPS*** *mode*

2.9.2. Network operation

Network operation runs on the background of the user program asynchronously with the cycle time of the user program. Nevertheless, current data exchange with the scratchpad takes place always during the I/O scan of the user program.

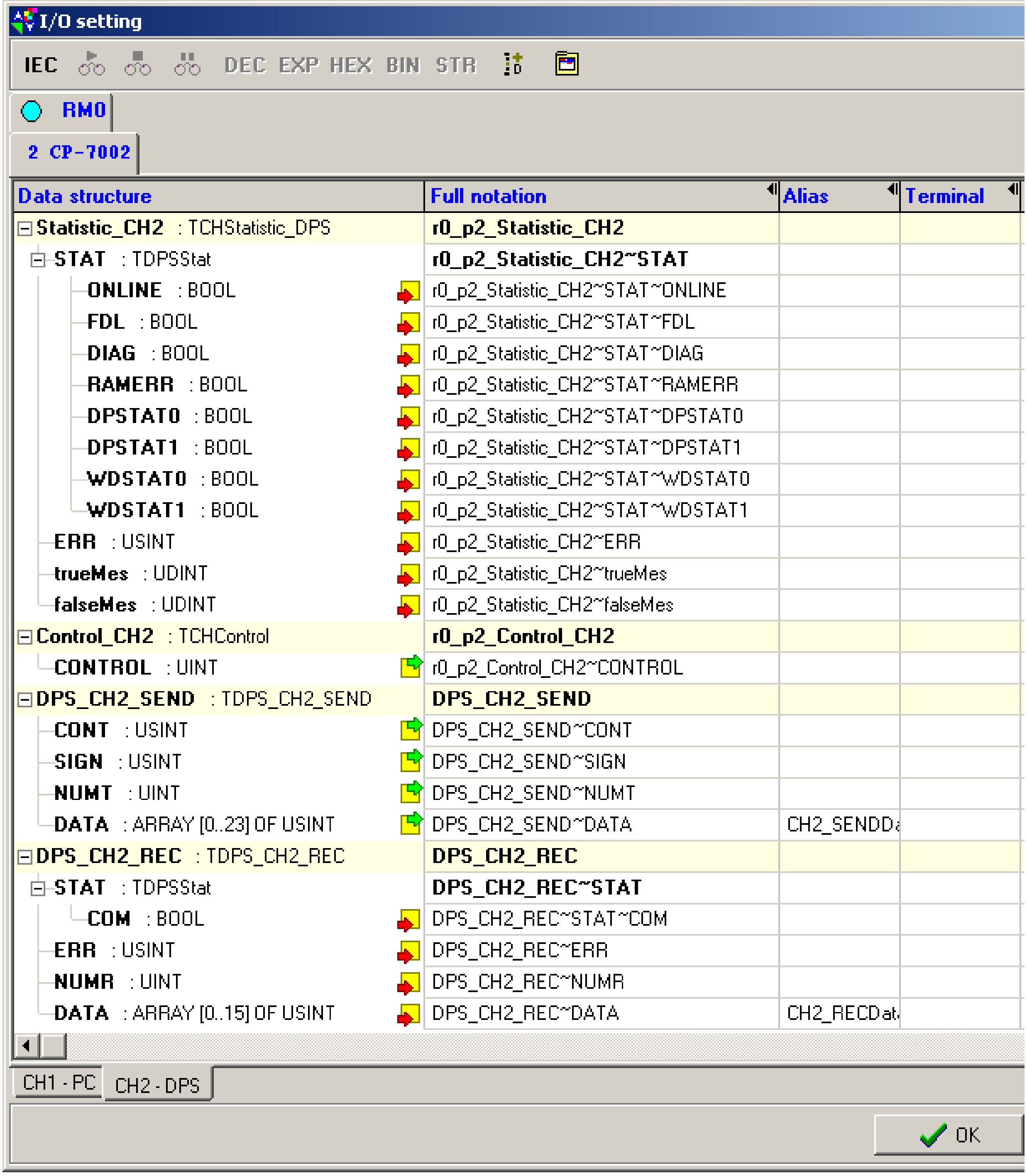
Operation diagnostics and network data

The communication channel at the **DPS** mode releases the diagnostics data of communication status with the PROFIBUS DP network, data received from the master station and data sent to the master station. These data are stored in the scratchpad and are easily accessible via the panel *I/O setting* through the icon IO (fig. 2.9.2).

Data have assigned symbolic names that begin with the rack and position number. In the column *Full notation* a concrete symbolic name for the given item is always specified.

If we want to use data in the user program, we use either this symbolic name or we enter our symbolic name in the column *Alias* which can be used then. In no case do we use absolute operands since they can alter after a new compilation of the user program.

Before processing of received data, it is necessary to check in the status the COM bit which determines data validity.



Data structure	Full notation	Alias	Terminal
Statistic_CH2 : TCHStatistic_DPS	r0_p2_Statistic_CH2		
STAT : TDPSStat	r0_p2_Statistic_CH2~STAT		
ONLINE : BOOL	r0_p2_Statistic_CH2~STAT~ONLINE		
FDL : BOOL	r0_p2_Statistic_CH2~STAT~FDL		
DIAG : BOOL	r0_p2_Statistic_CH2~STAT~DIAG		
RAMERR : BOOL	r0_p2_Statistic_CH2~STAT~RAMERR		
DPSTAT0 : BOOL	r0_p2_Statistic_CH2~STAT~DPSTAT0		
DPSTAT1 : BOOL	r0_p2_Statistic_CH2~STAT~DPSTAT1		
WDSTAT0 : BOOL	r0_p2_Statistic_CH2~STAT~WDSTAT0		
WDSTAT1 : BOOL	r0_p2_Statistic_CH2~STAT~WDSTAT1		
ERR : USINT	r0_p2_Statistic_CH2~ERR		
trueMes : UDINT	r0_p2_Statistic_CH2~trueMes		
falseMes : UDINT	r0_p2_Statistic_CH2~falseMes		
Control_CH2 : TCHControl	r0_p2_Control_CH2		
CONTROL : UINT	r0_p2_Control_CH2~CONTROL		
DPS_CH2_SEND : TDPS_CH2_SEND	DPS_CH2_SEND		
CONT : USINT	DPS_CH2_SEND~CONT		
SIGN : USINT	DPS_CH2_SEND~SIGN		
NUMT : UINT	DPS_CH2_SEND~NUMT		
DATA : ARRAY [0..23] OF USINT	DPS_CH2_SEND~DATA	CH2_SENDDa	
DPS_CH2_REC : TDPS_CH2_REC	DPS_CH2_REC		
STAT : TDPSStat	DPS_CH2_REC~STAT		
COM : BOOL	DPS_CH2_REC~STAT~COM		
ERR : USINT	DPS_CH2_REC~ERR		
NUMR : UINT	DPS_CH2_REC~NUMR		
DATA : ARRAY [0..15] OF USINT	DPS_CH2_REC~DATA	CH2_RECDat	

*Fig.2.9.2 Communication channel data in the **DPS** mode*

Network diagnostics:
Input data *rx_py_Statistic_XXX* (*TCHStatistic_DPS* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Stat - communication status (sets submodule MR-0152) (8-times bool type)

	WDSTAT1	WDSTAT0	DPSTAT1	DPSTAT0	RAMERR	DIAG	FDL	ONLINE
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

ONLINE - indication of connection of the MR-0152 submodule to the PROFIBUS DP network

0 -offline status (submodule not connected)
1 - online status (submodule connected)
FDL - saving of FDL indication (internal status of the ASIC circuit)
DIAG - diagnostic buffer status (internal status of the ASIC circuit)
RAMERR - error of access to the internal RAM of the submodule
0 - no error
1 - access error
DPSTAT1, DPSTAT0 - status of communication with master station
00 - status Wait_Prm (waiting for parameterisation)
01 - status Wait_Cfg (waiting for configuration)
10 - status Data_Exch (data exchange)
(the LED RxD of the relevant communication channel is permanently on)
11 - not occupied
WDSTAT1, WDSTAT0 - mode of internal timer Watchdog
00 - status Baud_Search (search for speed)
01 - status Baud_Control (speed check)
10 - status DP_Control (check of PROFIBUS DP operation)
11 - not occupied

Err - communication error (see chapter 2.9.3) (usint type)
trueMes - not used (udint type)
falseMes - not used (udint type)

Output data *rx_py_Control_XXX* (*TCHControl* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Control - not used (uint type)

Data sent to the master station *DPS_XXX_SEND* (*TDPS_ XXX_ SEND* structure):

(XXX - communication channel designation)

Cont - communication control (not used so far) (usint type)
Sign - reserve (usint type)
NumT - length of sent data (uint type)
Data [x] - sent data (usint array element type)

Data received from the master station *DPS_XXX_REC* (*TDPS_ XXX_ REC* structure):

(XXX - communication channel designation)

Stat - communication status with a participant (8-times bool type)

	X	X	X	X	X	X	COM	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

COM - communication status
0 - communication not established, following data are not valid
1 - communication established, following data valid

Err - not used (usint type)
NumR - length of received data (uint type)

Data [x] - received data (usint array element type)

Attention! If, in the master station, there is not the Watchdog activated in the parameterization of the relevant slave station then **the communication interruption is not detected!** The bit *COM* is set to log.1 at the moment of data exchange initialization and remains at this state until another initialization even in case of communication interruption. The LED diode RxD of the relevant communication channel behaves similarly.

Therefore, we strongly recommend to activate the Watchdog whenever!

The state when the Watchdog is not activated is, on the part of the slave station, recognised so that WDSTAT bits have even during functioning communication the value 01 - status Baud_Control while they should have the value 10 - status DP_Control.

Detailed information on the structure of individual telegrams and behaviour of PLC as the PROFIBUS DP slave station important to the programmers of master stations can be found in the manual named PLC TECOMAT as PROFIBUS DP slave station, TXV 004 05.02.

Operation indication

The LED diode RxD of the relevant serial channel serves as the indicator of the PROFIBUS DP network operation. If the station is in the user data exchange mode (Data_Exch), the LED is permanently on.

Within the data level the status of communication can be accurately found from the variable *Stat* in the diagnostic zone. After switching the PLC on, the initial value of this variable is 0.

Within the user program the MR-0152 submodule is initiated and connected to the PROFIBUS DP network. This is indicated by transition from the offline mode to the online mode (bit *Stat.ONLINE* = 1). The submodule monitors the network and sets the correct transmission rate and the watchdog merge from the status Baud_Search into the status Baud_Control (bits *Stat.WDSTAT1,0* = 01). At the moment of communication establishment with the master station, the watchdog switches to status DP_Control (bits *Stat.WDSTAT1,0* = 10). After a successful parameterisation the submodule switches from the status Wait_Prm to the status Wait_Cfg (bits *Stat.DPSTAT1,0* = 01) and after a successful configuration the submodule finally switches to the status Data_Exch (bits *Stat.DPSTAT1,0* = 10). If everything is OK and data exchange is performed, variable *Stat* has in the diagnostics zone the value of $A1.

An overview of states is given in table 2.9.1.

Table 2.9.1 States of communication in the PROFIBUS DP network reported by the MR-0152 submodule in the variable *Stat* in the diagnostics zone

Status	**Description**	**Value *Stat* (hex)**
offline	submodule not initiated	00
online, Baud_Search	transmission rate detection	01
Baud_Control	transmission rate detection	41
DP_Control, Wait_Prm	waiting for parameterisation	81
Wait_Cfg	waiting for configuration	91
Data_Exch	user data exchange	A1

2.9.3. Error messages

The operational error messages of the MR-0152 submodule are displayed in the variable *Err* in the diagnostic zone. The cause of these messages can be either a setup error on the side of the master station or a MR-0152 submodule failure.

1 ASIC circuit did not switch to offline mode, initialisation not possible

MR-0152 submodule failure.

2 ASIC circuit memory range exceeded

Too many requirements on the memory. The volume of data transmitted must be reduced.

3 Maximum data length exceeded

Invalid length of received or sent data specified.

6 maximum configuration length exceeded

The master station sent configuration data with a length exceeding the specified maximum (32 bytes).

Further information on the communication status can be found in the variable *Stat* in the diagnostic zone (see chapter 2.9.2.).

2.10. CAN MODE - CANOPEN STATIONS CONNECTION TO PLC

What is CANopen

CANopen is a data bus designated for data transmission in industrial environments among independent stations. The bus is based on the CAN interface and is designed especially for interconnection of logic units, intelligent sensors and actuators, often with their autonomous function. The CANopen bus is defined by the standard DS301 of the CiA association and is supported by a number of producers of various peripheral modules which usually have a high protection class, thus, can be used directly in technologies. The transmission rate is 500, 250, 125, 50, 20 or 10 kBd. A message protocol has a multiple data security ensuring that data successfully transmitted are correct.

CAN mode function

The connection of CANopen stations to the TECOMAT PLC is undertaken using the MR-0151 submodule. The data exchange between the submodule and the PLC is performed in the **CAN** mode. The **CAN** mode can be set only when the MR-0151 submodule is fitted in, otherwise, the channel is automatically switched off and an error during channel initialisation occurs. When the MR-0151 module is fitted, only the **CAN, CAS**, **CAB** or **OFF** mode can be set.

Type of modules mounted can be found in the Mosaic development environment through the option *PLC* | *HW configuration* in the tag *Module info* (see chapter 1.1.).

The difference between the **CAS** and the **CAN** mode (see chapter 2.11) is that the **CAN** mode performs active data exchange among the PLC and CANopen stations, so, strictly speaking it behaves as the master, while in the **CAS** mode data are only released to the CANopen network. If we want to implement more TECOMAT PLCs in the CANopen network, one PLC must be in the **CAN** mode and the other PLCs in the **CAS** mode.

CAB mode (see chapter 2.12.) is designated firstly for connection to the general CAN network, nevertheless, even here basic function of CANopen bus can be realized.

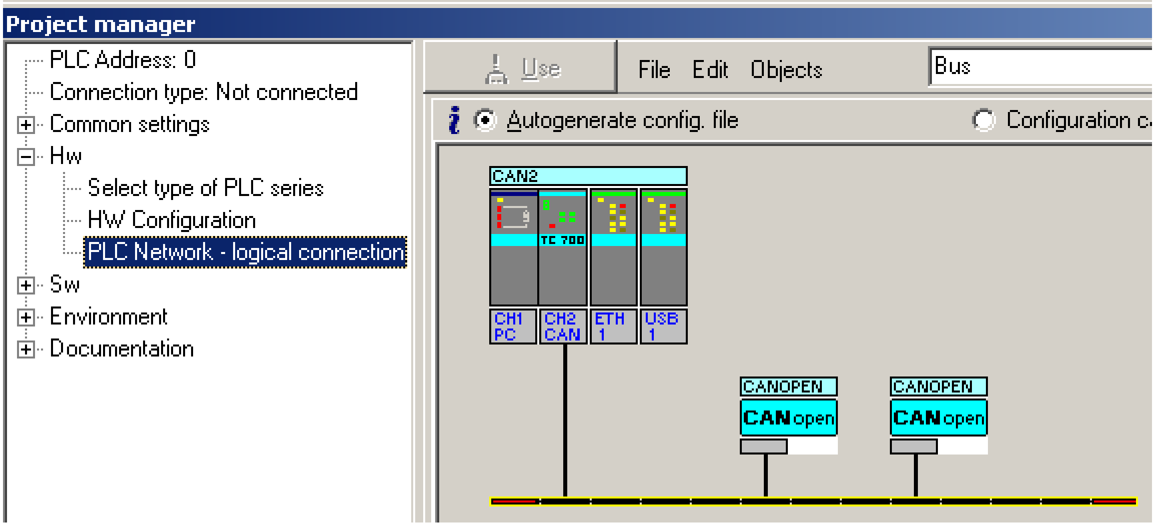
2.10.1. CANopen network initialisation

In the Mosaic development environment we select, in the Project manager, the folder *Hw* | *HW Configuration*. In the list of modules we highlight, using the mouse, the central unit, the channel of which we want to set. Then we press the push-button *Setting* or the icon ☑ on the line on the right from the position number. The panel *Channel parameters setting* (fig. 1.1) will be displayed. Set the corresponding serial channel to the **CAN** mode.

To be able to set all parameters of the CANopen network, we select the folder *Hw* | *PLC network* in the Project manager. Insert the PLC from the required projects on the desktop and insert the required number of further CANopen stations on the desktop using the option *Objects* | *CANopen.* Draw the bus (option *Objects* | *Bus*) and connect all CANopen stations and PLC channels in the **CAN** mode to it (fig. 2.10.1). Using right mouse click on the selected CANopen station and a local menu appears where you select the option *Network setting*. A table appears now where you can set all the parameters for data exchange with the selected CANopen station (fig. 2.10.2).

Stations within the CANopen network release their data primarily in so-called process data objects (PDOs) with a fixed length of 8 bytes each. These communication are labelled

TPDO1, TPDO2, TPDO3, TPDO4 (objects transmitted by the specified station) and RPDO1, RPDO2, RPDO3, RPDO4 (objects received by the specified station). In addition, the stations can use so-called service data objects (SDOs) allowing parameterisation of stations or usage of data exchange which is not time-critical.

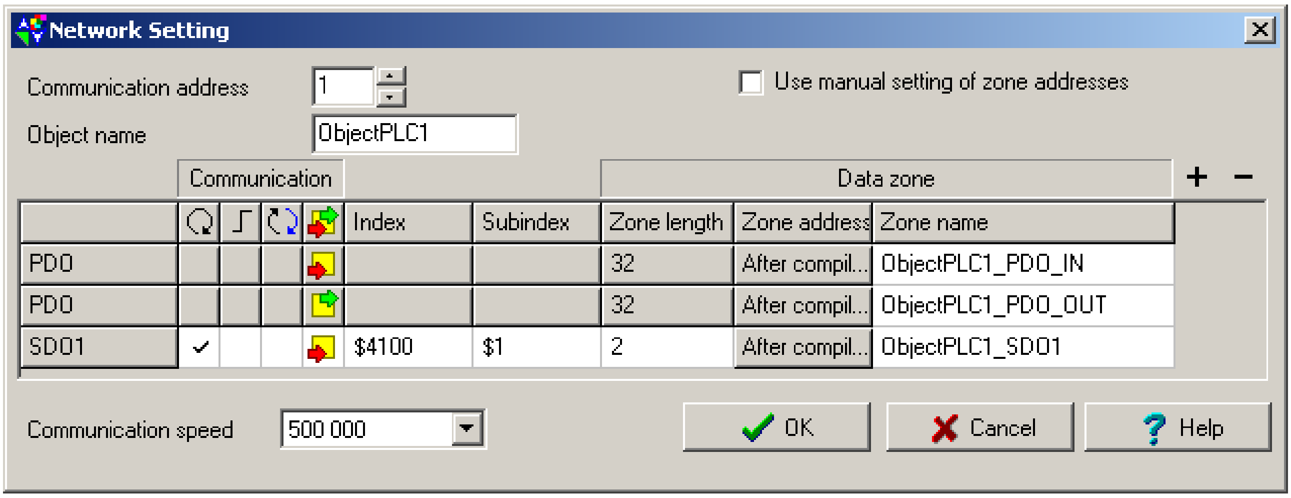


Project manager

- PLC Address: 0
- Connection type: Not connected
- Common settings
- Hw
  - Select type of PLC series
  - HW Configuration
  - PLC Network - logical connection
- Sw
- Environment
- Documentation

● Autogenerate config. file ○ Configuration c

Fig.2.10.1 CANopen network creation



Network Setting

Communication address: 1

Object name: ObjectPLC1

☐ Use manual setting of zone addresses

	Communication						Data zone		
					Index	Subindex	Zone length	Zone address	Zone name
PDO							32	After compil...	ObjectPLC1_PDO_IN
PDO							32	After compil...	ObjectPLC1_PDO_OUT
SDO1	✓				$4100	$1	2	After compil...	ObjectPLC1_SDO1

Communication speed: 500 000

Fig.2.10.2 Data exchange with CANopen station parameters setup

Following parameters are set in the table:

Communication address - station address (Node-ID) providing required data objects within the range of 1 to 31

Object name - the name that will become a part of the symbolic naming of data zones, used to distinguish individual stations.

PDO data objects are always transmitted. We can change the *Zone name* where we can enter the field names for input and output data, where TPDO1, TPDO2, TPDO3 and TPDO4, or RPDO1, RPDO2, RPDO3 and RPDO4 rather are positioned one after one. If we do not want the compiler to assign the *Zone address* automatically but we want to specify it by ourselves (e.g. because of connection to communication with another system), we have to enable the option *Use manual setting of zone addresses* in the top right corner of the table.

If we want to add a SDO object, press the "**+**" sign in the top right corner of the table. A new SDO line will be added in which we can set the following parameters:

Communication - method of SDO communication

- ↻ - periodic transmission of the SDO service data object
- ⎍ - unrepeated transmission of SDO controlled by the flag from the user program
- ⟳ - unrepeated transmission of SDO controlled by the flag from the user program or controlled automatically always when the station is not in operating mode (see chapter 2.10.2., section Station initialisation)
- ⇦ - servicing data object SDO will be read
- ⇨ - servicing data object SDO will be entered

Index - index SDO

Subindex - sub-index SDO

Zone length - object length SDO in bytes, maximum 4 bytes for reading and 64 bytes for writing

Also for an SDO object, we can change the *Zone name* where we can enter the field name for the data. If we do not want the compiler to assign the *Zone address* automatically but we want to specify it by ourselves (e.g. because of connection to communication with another system), we have to enable the option *Use manual setting of zone addresses* in the upper right corner of the table. An SDO object can be cancelled by highlighting using the mouse, followed by pressing the "**–**" sign in the top right corner of the table.

Communication speed - selection of transmission rate valid for the entire CANopen network

For each station it is possible to define one transmission of PDO objects and several SDO transmissions. In total, a maximum of 64 SDO object transmissions for all stations can be defined.

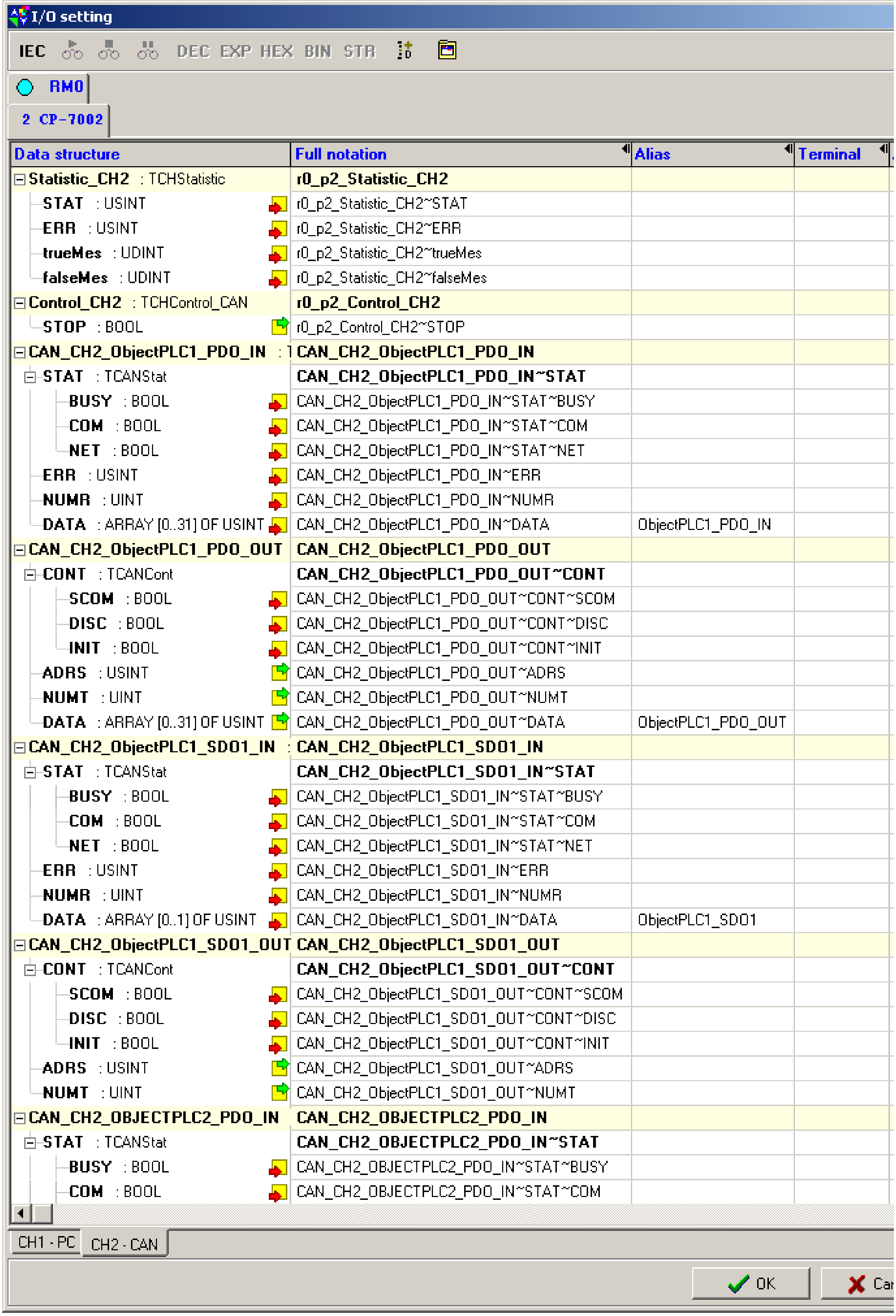
2.10.2. Network operation

The CANopen network operation is performed on the background of the user program asynchronously with the cycle time of the user program. All current network data are stored in the internal memory of the system and the data exchange with the scratchpad takes place always during the user program I/O scan.

Operation diagnostics and network data

The serial channels in the **CAN** mode release diagnostics data of network status, the data exchanged with another participants and the status of communication with these participants. These data are stored in the scratchpad and are easily accessible via the panel *I/O setting* through the icon IO (fig. 2.10.3).

Data have assigned symbolic names which begin with the rack and position number. In the column *Full notation*, a concrete symbolic name for the given item is always specified. If we want to use data in the user program, we use either this symbolic name, or we enter our symbolic name, which we then can use, in the column *Alias*. In no case do we use absolute operands since they can alter after a new compilation of the user program.



Data structure	Full notation	Alias	Terminal
⊟Statistic_CH2 : TCHStatistic	r0_p2_Statistic_CH2		
STAT : USINT	r0_p2_Statistic_CH2~STAT		
ERR : USINT	r0_p2_Statistic_CH2~ERR		
trueMes : UDINT	r0_p2_Statistic_CH2~trueMes		
falseMes : UDINT	r0_p2_Statistic_CH2~falseMes		
⊟Control_CH2 : TCHControl_CAN	r0_p2_Control_CH2		
STOP : BOOL	r0_p2_Control_CH2~STOP		
⊟CAN_CH2_ObjectPLC1_PDO_IN : T	CAN_CH2_ObjectPLC1_PDO_IN		
⊟STAT : TCANStat	CAN_CH2_ObjectPLC1_PDO_IN~STAT		
BUSY : BOOL	CAN_CH2_ObjectPLC1_PDO_IN~STAT~BUSY		
COM : BOOL	CAN_CH2_ObjectPLC1_PDO_IN~STAT~COM		
NET : BOOL	CAN_CH2_ObjectPLC1_PDO_IN~STAT~NET		
ERR : USINT	CAN_CH2_ObjectPLC1_PDO_IN~ERR		
NUMR : UINT	CAN_CH2_ObjectPLC1_PDO_IN~NUMR		
DATA : ARRAY [0..31] OF USINT	CAN_CH2_ObjectPLC1_PDO_IN~DATA	ObjectPLC1_PDO_IN	
⊟CAN_CH2_ObjectPLC1_PDO_OUT	CAN_CH2_ObjectPLC1_PDO_OUT		
⊟CONT : TCANCont	CAN_CH2_ObjectPLC1_PDO_OUT~CONT		
SCOM : BOOL	CAN_CH2_ObjectPLC1_PDO_OUT~CONT~SCOM		
DISC : BOOL	CAN_CH2_ObjectPLC1_PDO_OUT~CONT~DISC		
INIT : BOOL	CAN_CH2_ObjectPLC1_PDO_OUT~CONT~INIT		
ADRS : USINT	CAN_CH2_ObjectPLC1_PDO_OUT~ADRS		
NUMT : UINT	CAN_CH2_ObjectPLC1_PDO_OUT~NUMT		
DATA : ARRAY [0..31] OF USINT	CAN_CH2_ObjectPLC1_PDO_OUT~DATA	ObjectPLC1_PDO_OUT	
⊟CAN_CH2_ObjectPLC1_SDO1_IN :	CAN_CH2_ObjectPLC1_SDO1_IN		
⊟STAT : TCANStat	CAN_CH2_ObjectPLC1_SDO1_IN~STAT		
BUSY : BOOL	CAN_CH2_ObjectPLC1_SDO1_IN~STAT~BUSY		
COM : BOOL	CAN_CH2_ObjectPLC1_SDO1_IN~STAT~COM		
NET : BOOL	CAN_CH2_ObjectPLC1_SDO1_IN~STAT~NET		
ERR : USINT	CAN_CH2_ObjectPLC1_SDO1_IN~ERR		
NUMR : UINT	CAN_CH2_ObjectPLC1_SDO1_IN~NUMR		
DATA : ARRAY [0..1] OF USINT	CAN_CH2_ObjectPLC1_SDO1_IN~DATA	ObjectPLC1_SDO1	
⊟CAN_CH2_ObjectPLC1_SDO1_OUT	CAN_CH2_ObjectPLC1_SDO1_OUT		
⊟CONT : TCANCont	CAN_CH2_ObjectPLC1_SDO1_OUT~CONT		
SCOM : BOOL	CAN_CH2_ObjectPLC1_SDO1_OUT~CONT~SCOM		
DISC : BOOL	CAN_CH2_ObjectPLC1_SDO1_OUT~CONT~DISC		
INIT : BOOL	CAN_CH2_ObjectPLC1_SDO1_OUT~CONT~INIT		
ADRS : USINT	CAN_CH2_ObjectPLC1_SDO1_OUT~ADRS		
NUMT : UINT	CAN_CH2_ObjectPLC1_SDO1_OUT~NUMT		
⊟CAN_CH2_OBJECTPLC2_PDO_IN	CAN_CH2_OBJECTPLC2_PDO_IN		
⊟STAT : TCANStat	CAN_CH2_OBJECTPLC2_PDO_IN~STAT		
BUSY : BOOL	CAN_CH2_OBJECTPLC2_PDO_IN~STAT~BUSY		
COM : BOOL	CAN_CH2_OBJECTPLC2_PDO_IN~STAT~COM		

*Fig.2.10.3 Serial channel data in the **CAN** mode*

Into zones with output data the user enters data to be written in corresponding slave stations. Data transmitted from corresponding slave stations are in zones with input data. Before they are processed, it is necessary to check the COM bit in the relevant status.

Network diagnostics:
Input data *rx_py_Statistic_XXX* (*TCHStatistic* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Stat - communication status (not used for now) (usint type)
Err - communication error (see chapter 2.10.3) (usint type)
trueMes - number of valid network communications (not used for now) (udint type)
falseMes - number of invalid network communications (not used for now) (udint type)

Output data *rx_py_Control_XXX* (*TCHControl_CAN* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Control - communication control (uint type)

	X	X	X	X	X	X	X	STOP
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

	X	X	X	X	X	X	X	X
bit	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*

STOP - network control
0 - station in operation mode
1 - all stations in STOP mode, outputs blocked (providing stations support this status)

Data exchanged with the participant:
Input data PDO *CAN_XXX_NNN_PDO_IN* (*TCAN_XXX_NNN_PDO_IN* structure):

(XXX - communication channel designation, NNN - object name set according to fig. 2.10.2)

Stat - communication with participant status (8-times bool type)

	X	X	X	X	X	X	COM	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

COM - communication status
0 - station is not in operation mode, following data not valid
1 - station is in operation mode, following data valid

Err - communication with participant error (see chapter 2.10.3) (usint type)
NumR - length of read data from the participant (uint type)
Data [x] - read data from participant (usint array element type)

Output data PDO *CAN_XXX_NNN_PDO_OUT* (*TCAN_XXX_NNN_PDO_ OUT* structure):

(XXX - communication channel designation, NNN - object name set according to fig. 2.10.2)

Cont - communication control (8-times bool type)

	INI	0	0	0	0	0	0	0
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

INI - station initialisation execution
0 - station initialisation is not executed
1 - complete station initialisation execution

Sign - reserve (type usint)
NumT - length of entered data from participant (uint type)
Data [x] - data entered from participant (usint array element type)

Input data SDO *CAN_XXX_NNN_SDOn_IN* (*TCAN_XXX_NNN_SDOn_IN* structure):

(XXX - communication channel designation, NNN - object name set according to fig. 2.10.2, n - SDO number)

Stat - communication status with the participant (8-times bool type)

	X	X	X	X	X	X	COM	BUSY
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

COM - communication status
0 - station is not in operation mode, following data not valid
1 - station is in operating mode, following data valid
BUSY- unrepeated communication status
0 - unrepeated communication does not proceed
1 - unrepeated communication proceeds, data not loaded yet

Err - communication with participant error (see chapter 2.10.3) (usint type)
NumR - length of read data from participant (uint type)
Data [x] - data read from participant (usint array element type)

Output data SDO *CAN_XXX_NNN_SDOn_OUT* (*TCAN_XXX_NNN_SDOn_OUT* structure):

(XXX - communication channel designation, NNN - object name set according to fig. 2.10.2, n - SDO number)

Cont - communication control (8-times bool type)

	INI	0	0	0	0	0	0	SCOM
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

INI - station initialisation execution
0 - station initialisation not executed
1 - complete station initialisation execution
SCOM - unrepeated communication execution (for unrepeated and initialisation SDO only)
0 - unrepeated communication will not proceed
1 - unrepeated communication activation

Sign	- reserve (usint type)
NumT	- length of written data from participant (uint type)
Data [x]	- data written from participant (usint array element type)

Cyclic data exchange

The PDO and periodic SDO object data are continually refreshed according to the parameters specified in the initialisation. Here must be emphasized that SDO data transmission is significantly slower, in comparison to the PDO data transmission. Before the read data is processed, it is necessary to check the COM bit in the relevant status. If it has the value of log.1, the station is in operating mode and data are valid.

Unrepeated data exchange

If SDO objects are declared as unrepeated or initialisation ones, communication is started after setting the SCOM bit in the control byte to log.1. Immediately before enqueuing of this communication in the list, the SCOM bit is set to zero and the BUSY bit in the status is set to log.1. After performing communication the BUSY bit is set to zero and the COM bit is set according to the result of communication. If it has the value of log.1, data are valid. In the relevant part of the communication zone of the master system, data are unrepeatedly transmitted from the station, if it was required.

In practise, it means that we set the SCOM bit to log.1 and, in the following cycle of the user program, we test the BUSY bit. If it is log.1, we repeat the test in the following cycle. If the BUSY bit has the value of log.0, we check the COM bit. If the COM bit is log.1, data exchange was undertaken without any error and any possible data received are valid.

Station initialisation

Some CANopen stations require setting of several parameters to be able to switch to the operation mode. For this purpose, SDO communication objects are used which are declared as initialisation ones.

After restarting the PLC user program, all communications which are declared as initialisation ones with SDO objects are executed. Communication run in the exact order, in which the SDOs are entered to initialisation tables of the CANopen network (fig. 2.10.2). Afterwards, stations are switched to the operating mode and periodic data exchange within PDO objects and periodic SDOs proceeds.

If the CANopen station is not in the operating mode and communication with it is running (e.g. a short-term station power supply failure), all communication with initialisation SDOs assigned to this station (according to the address) are automatically executed in the order, in which they were declared. The initialisation of the CANopen station can be initiated any time simply by setting the INI bit in the control byte of any communication assigned to this station to log.1.

The second possibility, how the CANopen station can be initiated, is by downloading the initialisation, supplied by the station producer, to the EEPROM memory of this station. It is not necessary to declare the initialisation SDO in the PLC user program then.

Bus time parameters

The transmission of TPDO1 and RPDO1 of all stations with address 1 to 31 is ensured in a maximum 30 ms raster, independently of the number of stations. The transmission of other PDOs depends on the maximum address of the station connected (see table 2.10.1). SDO objects are transmitted successively as they are declared and activated (in case of unrepeated entries). The transmission of one SDO takes about 5 to 10 ms, depending on station response (timeout is 20 ms). SDO and PDO transmissions are not time-affected by each other.

Cycle guard is 70 ms.

Table 2.10.1 Time parameters of PDO transmission

Used station addresses in the CANopen network	TPDO1, RPDO1	TPDO2, TPDO3, TPDO4, RPDO2, RPDO3, RPDO4
1 - 5	30 ms	30 ms
1 - 9	30 ms	60 ms
1 - 14	30 ms	90 ms
1 - 18	30 ms	120 ms
1 - 22	30 ms	150 ms
1 - 26	30 ms	180 ms
1 - 31	30 ms	210 ms

A recommendation follows from the table to address stations ascendently from 1 so that the network operation is optimal.

2.10.3. Error messages

In the *Err* variable of the receiving zone of each slave station, a communication error code is stored.

$08 communication timeout

The station did not respond to the SDO transmission within the specified period of time. It is necessary to check the physical connection of the station and its communication parameters settings (address and transmission rate).

2.11. CAS MODE - CANopen STATION REALIZATION

What is CANopen

CANopen is a data bus designated for data transmission in the industrial environment among independent stations. The bus is based on the CAN interface and is designed especially for interconnection of logic units, intelligent sensors and actuators, often with their autonomous function. The CANopen bus is defined by the standard DS301 of the CiA association and is supported by a number of producers of various peripheral modules which usually have a high protection class and thus, they can be used directly within technologies. The transmission rate is 500, 250, 125, 50, 20 or 10 kBd. A message protocol has a multiple data security ensuring that data successfully transmitted are correct.

CAS mode function

CANopen station implementation in the TECOMAT PLC is undertaken using the MR-0151 submodule. The data exchange between the submodule and PLC is performed in the **CAS** mode.

The **CAS** mode can be set only when the MR-0151 submodule is fitted, otherwise, the channel is switched off automatically and error occurs during channel initialisation. When the MR-0151 module is fitted, the **CAN, CAS, CAB** or **OFF** mode can be set. The type of modules fitted can be found in the Mosaic development environment via the option *PLC | HW configuration* in the tag *Module info* (see chapter 1.1.).

The difference between the **CAS** and the **CAN** mode (see chapter 2.10.) is that in the basic mode, so-called slave mode (see chapter 2.11.1.1), the CAS mode releases data within the CANopen network but does not participate actively on communication, so, strictly speaking, it behaves as a slave, while the **CAN** mode performs active data exchange among the PLC and CANopen stations. If we want to implement more TECOMAT PLCs in the CANopen network, one PLC must be in the **CAN** mode and the other PLCs in the **CAS** mode.

By means of the **CAS** mode, it is possible to implement a special case of PLC interconnection through the CANopen network where it is possible to change addresses of individual stations and to assign the master function to one of them from the user program running at that time.

All these stations are in the **CAS** mode which has set a so-called **floating master mode** (see chapter 2.11.1.2.). Such an interconnection is useful in cases when the PLC is a part of a mobile or plug-in unit and there is a request to disconnect and re-connect these units into other assemblies under run.

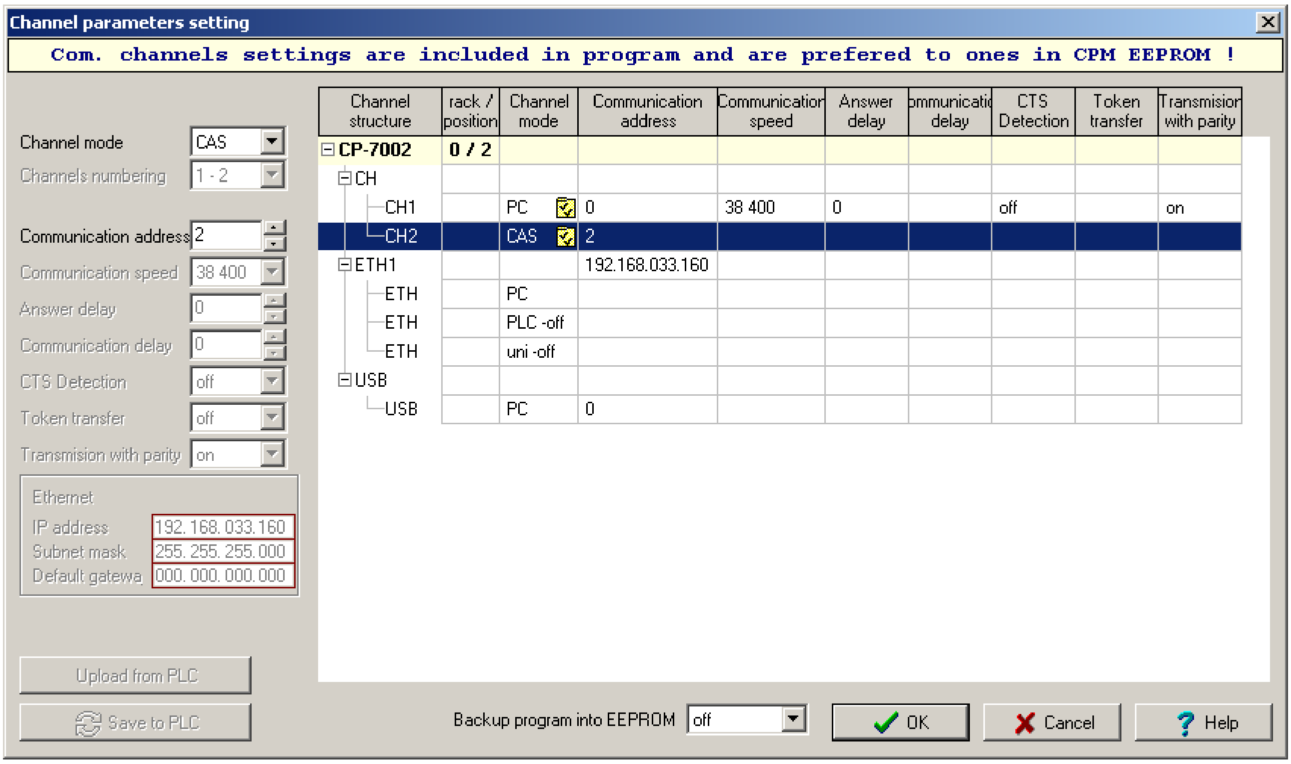
The **CAB** mode (see chapter 2.12) is designated mostly for interconnection to the general network CAN, nevertheless, even within this mode there can be realized basic function of the CANopen bus.

2.11.1. Parameters setup

In the Mosaic development environment in the Project manager, we select the folder *Hw | HW Configuration*. In the list of modules we highlight the central unit, the channel of which we want to set, using the mouse. Then we press the push-button *Setting*, or icon ☑ on the line on the right from the position number. The panel *Channel parameters setting* (fig. 2.11.1) will appear. Set the corresponding serial channel to the **CAS** mode.

To be able to set all parameters of the CANopen network, we select the folder *Hw* | *PLC network* in the Project manager.

Insert the PLC from the required projects onto the desktop and possibly, insert the required number of further CANopen stations onto the desktop using the option *Objects* | *CANopen.* Draw the bus (option *Objects* | *Bus*) and connect to it all CANopen stations and PLC channels in the **CAS** mode (fig. 2.11.2). Using the right-mouse click on the selected CANopen station or PLC channel a local menu appears where we select the option *Network setting*. A table appears where you can set all the parameters for data exchange (fig. 2.11.3, fig. 2.11.4).



Channel parameters setting

Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

Channel mode: CAS
Channels numbering: 1 - 2
Communication address: 2
Communication speed: 38 400
Answer delay: 0
Communication delay: 0
CTS Detection: off
Token transfer: off
Transmision with parity: on

Ethernet
IP address: 192. 168. 033. 160
Subnet mask: 255. 255. 255. 000
Default gatewa: 000. 000. 000. 000

Channel structure	rack / position	Channel mode	Communication address	Communicatior speed	Answer delay	ommunicatio delay	CTS Detection	Token transfer	Transmision with parity
⊟ CP-7002	0 / 2								
⊟ CH									
CH1		PC	0	38 400	0		off		on
CH2		CAS	2						
⊟ ETH1			192.168.033.160						
ETH		PC							
ETH		PLC -off							
ETH		uni -off							
⊟ USB									
USB		PC	0						

Backup program into EEPROM: off

Fig.2.11.1 ***CAS*** *mode setting*

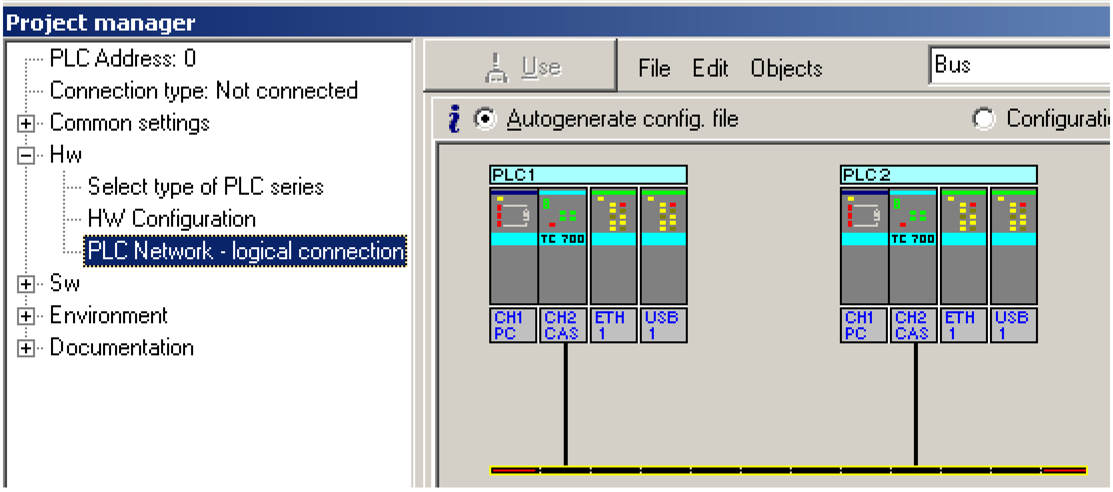


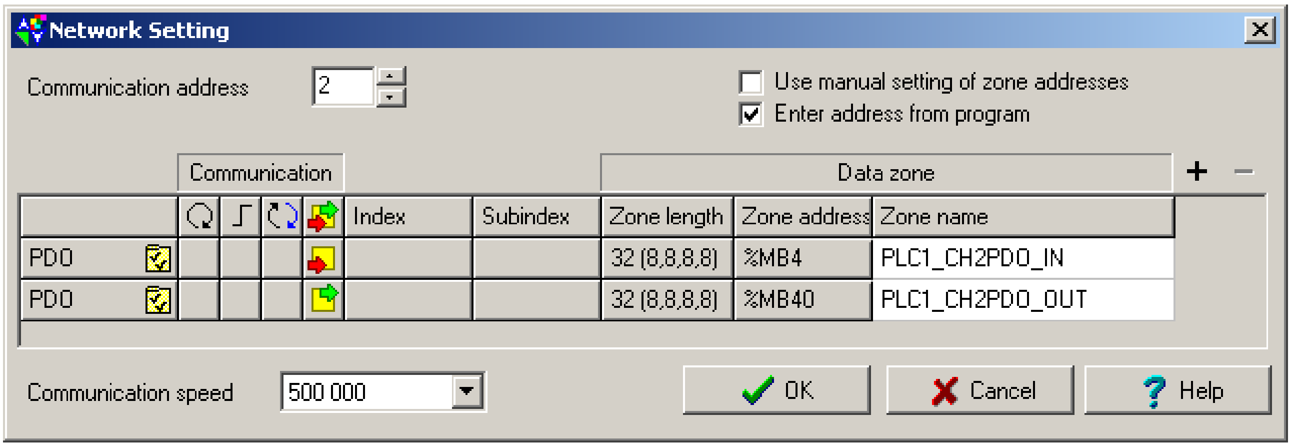
Fig. 2.11.2 PLC network in the ***CAS*** *mode*

If all connected PLCs in the network in the **CAS** mode are set to the slave mode, then **one PLC in this network must be** in the **CAN** mode so that then it controls the bus operation (it is permanently in the master status).

If at least one of the connected PLCs in the network in the **CAS** mode is set to the floating master mode, then **none of the PLCs must be set** to the **CAN** mode in this network. The bus operation and data transmission is performed by the PLC being just in the master status.

In both cases, also other CANopen stations that are always in the slave status can be connected to the bus.

2.11.1.1. Slave mode setup



Communication address 2

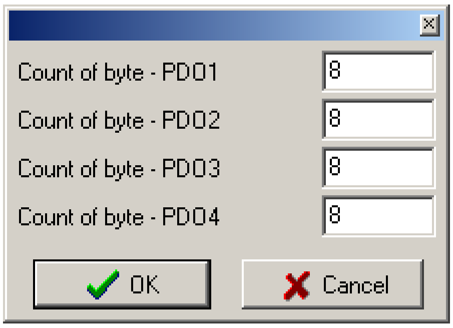
☐ Use manual setting of zone addresses

☑ Enter address from program

	Communication						Data zone		
					Index	Subindex	Zone length	Zone address	Zone name
PDO ☑							32 (8,8,8,8)	%MB4	PLC1_CH2PDO_IN
PDO ☑							32 (8,8,8,8)	%MB40	PLC1_CH2PDO_OUT

Communication speed 500 000

Fig.2.11.3 ***CAS - slave*** *mode parameters setting*



Count of byte - PDO1	8
Count of byte - PDO2	8
Count of byte - PDO3	8
Count of byte - PDO4	8

Fig.2.11.4 Individual PDOs lengths setup

In case the PLC should behave as a passive provider or receiver of data in the CANopen network, it is necessary to set only the location of received data (RxPDO1 - RxPDO4) and the location of sent data (TxPDO1 - TxPDO4). The fixed length of received or sent data is 32 bytes. Individual PDOs have a max length of 8 bytes that can be changed using a dialog box available through the icon ☑ in the first column of the relevant line of the table.

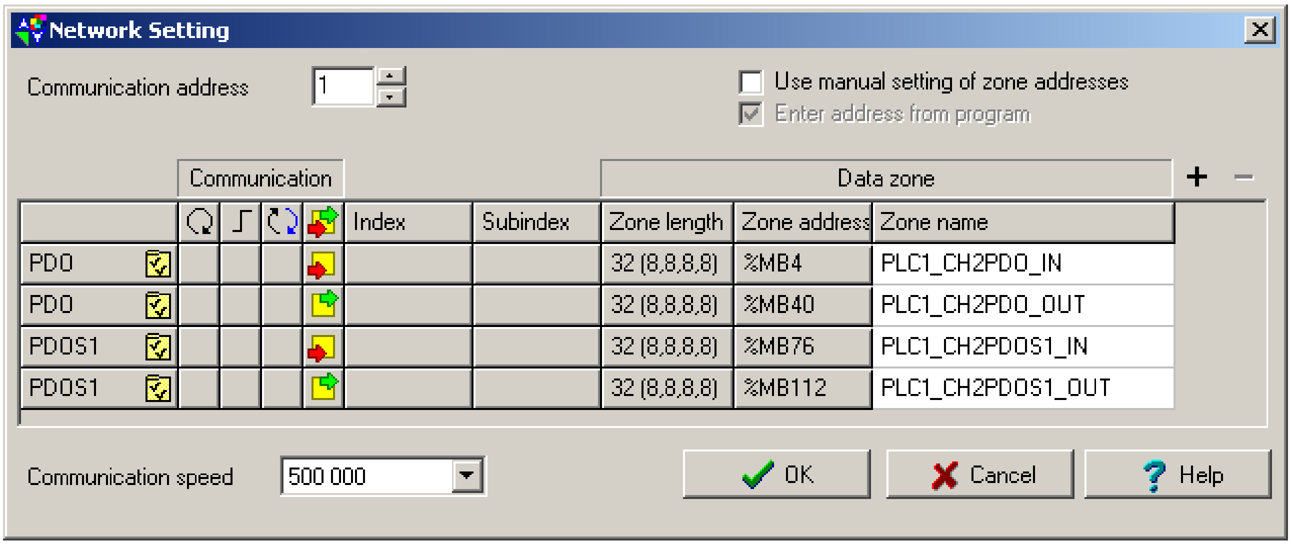
The dialog box appears (fig.2.11.4) that enables the setting of the length for all four PDOs in the range 0 to 8 (where 0 means transfer switched off). Actual values of sizes of individual PDOs are shown in brackets behind the length of the zone on the relevant line. Regardless of their size, PDOs are situated in communication zones always in 8bytes intervals (during their change, there will not occur their shift in the scratchpad).

The symbolic name (or the absolute address) refers to the received or sent data itself, preceded by four status or control bytes rather (see the data structure in chapter 2.11.2.). The physical address is automatically assigned by the compiler and it is displayed in the field *Zone address*. If you need to assign a certain physical address by yourself, enable the option *Use manual setting of zone addresses*. Then the field *Zone address* will become available for editing.

In the field *Communication address* set the station address (node address). If we need to change the station address under run, enable the option *Enter address from program*. The address station will be then received from variable *Adrs* in the structure of the output data, at setting of bit *REST* in the network control *Control* (chapter 2.11.2.). Then the value set in the field *Address for communication* has no meaning. The station address must not be identical with any address of any participant in the CANopen network and must not be 0.

A detailed description of the operation in the slave mode is given in chapter 2.11.2.1.

2.11.1.2. Floating master mode setup



Network Setting

Communication address 1

☐ Use manual setting of zone addresses
☑ Enter address from program

		Communication						Data zone		
						Index	Subindex	Zone length	Zone address	Zone name
PDO	☑				➡			32 (8,8,8,8)	%MB4	PLC1_CH2PDO_IN
PDO	☑				↗			32 (8,8,8,8)	%MB40	PLC1_CH2PDO_OUT
PDOS1	☑				➡			32 (8,8,8,8)	%MB76	PLC1_CH2PDOS1_IN
PDOS1	☑				↗			32 (8,8,8,8)	%MB112	PLC1_CH2PDOS1_OUT

Communication speed 500 000

Fig.2.11.5 ***CAS - floating master mode*** *parameters setting*

The floating master mode allows creating and changing the network configuration of the stations under run. If the station being programmed is to have the possibility to switch to the master status and to control the operation on the bus, then we have add as many output and input zones by means of the **+** push-button as the maximum number of connectable stations will be (7 is the maximum).

The fixed length of received or sent data is 32 bytes. Individual PDOs have a max length of 8bytes that can be changed using a dialog box available through the icon ☑ in the first column of the relevant line of the table. The dialog box appears (fig.2.11.4) that enables the setting of the length for all four PDOs in the range 0 to 8 (where 0 means transfer switched off). Actual values of sizes of individual PDOs are shown in brackets behind the length of the zone on the relevant line. Regardless of their size, PDOs are situated in communication zones always in 8bytes intervals (during their change, there will not occur their shift in the scratchpad).

The option *Enter address from program* is enabled automatically, since the address station is received from variable *Adrs* in the structure of the output data itself, at setting of bit *REST* in network control *Control* (chapter 2.11.2.). The value set in the field *Communication address* has no meaning. The station address must not be identical with any address of any participant of the CANopen network and must not be 0.

For each zone including the own one it is necessary to set the location of received data (own data RxPDO1 - RxPDO4, foreign data TxPDO1 - TxPDO4) and the location of sent data (own data TxPDO1 - TxPDO4, foreign data RxPDO1 - RxPDO4). The fixed length of received or sent data is 32 bytes.

The symbolic name (or the absolute address) refers to the received or sent data itself, preceded by four status or control bytes as the case may be (see data structure in chapter 2.11.2.). A detailed description of the operation is given in chapter 2.11.2.2.

The physical address is automatically assigned by the compiler and it is displayed in the field *Zone address*. If you need to assign a certain physical address by yourself, enable the option *Use manual setting of zone addresses*. Then the field *Zone address* will become available for editing.

2.11.2. Network operation

The network operation is performed on the background of the user program asynchronously with the cycle time of the user program. Nevertheless, the current data exchange with the scratchpad is always executed during the user program I/O scan.

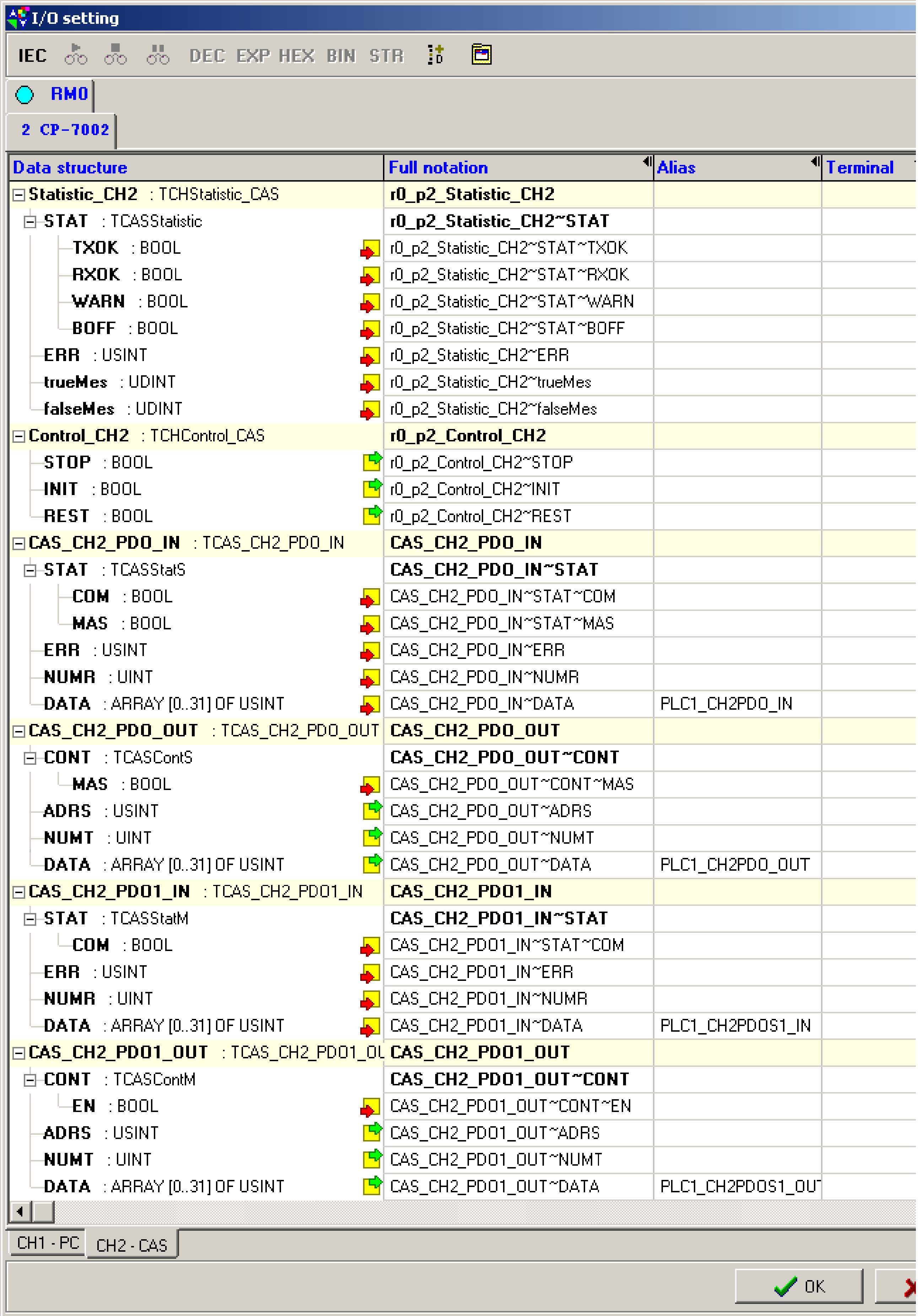
Operation diagnostics and network data

The communication channel in the **CAS** mode releases diagnostics data of the communication status with the CANopen network, received and sent data. Data are stored in the scratchpad and are easily accessible via the panel *I/O setting*, accessible through the icon IO (fig. 2.11.6).

Data have assigned symbolic names that begin with the rack and position number. In the column *Full notation*, a concrete symbolic name for the given item is always specified.

If we want to use data in the user program, we use either this symbolic name or we enter in the column *Alias* our symbolic name which can be used then. In no case do we use absolute operands, since they can alter after a new compilation of the user program.

Data processing in the slave mode is described in the chapter 2.11.2.1. Data processing in the floating master mode is described in the chapter 2.11.2.2.



Data structure	Full notation	Alias	Terminal
Statistic_CH2 : TCHStatistic_CAS	r0_p2_Statistic_CH2		
STAT : TCASStatistic	r0_p2_Statistic_CH2~STAT		
TXOK : BOOL	r0_p2_Statistic_CH2~STAT~TXOK		
RXOK : BOOL	r0_p2_Statistic_CH2~STAT~RXOK		
WARN : BOOL	r0_p2_Statistic_CH2~STAT~WARN		
BOFF : BOOL	r0_p2_Statistic_CH2~STAT~BOFF		
ERR : USINT	r0_p2_Statistic_CH2~ERR		
trueMes : UDINT	r0_p2_Statistic_CH2~trueMes		
falseMes : UDINT	r0_p2_Statistic_CH2~falseMes		
Control_CH2 : TCHControl_CAS	r0_p2_Control_CH2		
STOP : BOOL	r0_p2_Control_CH2~STOP		
INIT : BOOL	r0_p2_Control_CH2~INIT		
REST : BOOL	r0_p2_Control_CH2~REST		
CAS_CH2_PDO_IN : TCAS_CH2_PDO_IN	CAS_CH2_PDO_IN		
STAT : TCASStatS	CAS_CH2_PDO_IN~STAT		
COM : BOOL	CAS_CH2_PDO_IN~STAT~COM		
MAS : BOOL	CAS_CH2_PDO_IN~STAT~MAS		
ERR : USINT	CAS_CH2_PDO_IN~ERR		
NUMR : UINT	CAS_CH2_PDO_IN~NUMR		
DATA : ARRAY [0..31] OF USINT	CAS_CH2_PDO_IN~DATA	PLC1_CH2PDO_IN	
CAS_CH2_PDO_OUT : TCAS_CH2_PDO_OUT	CAS_CH2_PDO_OUT		
CONT : TCASContS	CAS_CH2_PDO_OUT~CONT		
MAS : BOOL	CAS_CH2_PDO_OUT~CONT~MAS		
ADRS : USINT	CAS_CH2_PDO_OUT~ADRS		
NUMT : UINT	CAS_CH2_PDO_OUT~NUMT		
DATA : ARRAY [0..31] OF USINT	CAS_CH2_PDO_OUT~DATA	PLC1_CH2PDO_OUT	
CAS_CH2_PDO1_IN : TCAS_CH2_PDO1_IN	CAS_CH2_PDO1_IN		
STAT : TCASStatM	CAS_CH2_PDO1_IN~STAT		
COM : BOOL	CAS_CH2_PDO1_IN~STAT~COM		
ERR : USINT	CAS_CH2_PDO1_IN~ERR		
NUMR : UINT	CAS_CH2_PDO1_IN~NUMR		
DATA : ARRAY [0..31] OF USINT	CAS_CH2_PDO1_IN~DATA	PLC1_CH2PDOS1_IN	
CAS_CH2_PDO1_OUT : TCAS_CH2_PDO1_OL	CAS_CH2_PDO1_OUT		
CONT : TCASContM	CAS_CH2_PDO1_OUT~CONT		
EN : BOOL	CAS_CH2_PDO1_OUT~CONT~EN		
ADRS : USINT	CAS_CH2_PDO1_OUT~ADRS		
NUMT : UINT	CAS_CH2_PDO1_OUT~NUMT		
DATA : ARRAY [0..31] OF USINT	CAS_CH2_PDO1_OUT~DATA	PLC1_CH2PDOS1_OU	

*Fig.2.11.6 Communication channel data in the **CAS** mode*

Network diagnostics:
Input data *rx_py_Statistic_XXX* (*TCHStatistic* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Stat - CAN bus status (8-times bool type)

	BOFF	WARN	X	RXOK	TXOK	X	X	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

TXOK - messages broadcasting on the CAN bus
0 - valid messages not sent
1 - valid messages sent

RXOK - messages receiving on the CAN bus
0 - valid messages not received
1 - valid messages received

WARN - errors occurance on the CAN bus
0 - excessive amount of errors not detected on the CAN bus
1 - excessive amount of errors occurred on the CAN bus, their number exceeded 96

BOFF - the connection of the sequencer to the CAN bus
0 - sequencer is connected to the CAN bus
1 - abnormal error occurance on the CAN bus, their number exceeded 256. The sequencer was disconnected from the bus (bus-off status) and requires a new initialization via INIT bit in the variable Control

Err - Communication error notified by the CAN sequencer (usint type)

0 - no error

1 - stuff error
More than 5 similar bits occurred in the sequence in the part of the received message where it is not allowed

2 - form error
The fixed format zone of the received message has a false format.

3 - acknowledgment error
Sent message was not confirmed by any nod.

4 - bit 1 error
During the message sending (excluding arbitrary array), the sequencer sent the bit in log.1 but the status of the bus was log.0.

5 - bit 0 error
During the message sending (excluding arbitrary array), the sequencer sent the bit in log.0 but the status of the bus was log.1. During the bus-off status this means a permanent error.

6 - CRC error
CRC error control character in the received message is false. .

trueMes - number of valid network communication (udint type)
falseMes - number of invalid network communication (udint type)

Output data *rx_py_Control_XXX* (*TCHControl_CAS* structure):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Control - communication control (uint type)

	REST	INIT	X	X	X	X	X	STOP
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

	X	X	X	X	X	X	X	X
bit	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*

STOP - network control (valid only in the master status)
0 - station in operation mode
1 - all stations in STOP mode, outputs blocked (providing stations support this status)

INIT - network restart with the sequencer initialization
0 - the network is operated with parameters specified during the last restart
1 - perform network restart according to new parameters in data zones including new initialization of the CAN sequencer (necessary when the sequencer is in the bus-off status; the bit is automatically deleted)

REST - network restart
0 - the network is operated with parameters specified during the last restart
1 - perform network restart according to new parameters in data zones (the bit is automatically deleted)

Exchanged data in the slave status with another master station:
Input data PDO *CAS_XXX_PDO_IN* (*TCAS_XXX_PDO_IN* structure):

(XXX - communication channel designation)

Stat - communication status with participant (8-times bool type)

	X	MAS	X	X	X	X	COM	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

COM - communication status
0 - communication not established, following data are not valid
1 - communication established, following data valid

MAS - operational status of this station
0 - slave status - data are exchanged with another master station
1 - master status - data exchange with slave stations through additional data zones

Err - not used (usint type)
NumR - length of received data (uint type)
Data [x] - received data (usint array element type) - valid only in the slave status

Output data PDO *CAS_XXX_PDO_OUT* (*TCAS_XXX_PDO_OUT* structure):

(XXX - communication channel designation)

Cont - communication control (usint type)

	X	MAS	X	X	X	X	X	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

MAS - selection of operating status of this station - valid only at the moment of the network restart (*Control.REST* = 1)
0 - slave status
1 - master status

Adrs - current address of this station (usint type) - valid only at the moment of the network restart (*Control.REST* = 1) in the slave mode when the option *Enter address from program* (fig. 2.11.2) is enabled or in the floating master mode

NumT - length of sent data (uint type)

Data [x] - sent data (usint array element type) - valid only in the slave status

Exchanged data in the master status with other slave stations:

(zones exist only within the floating master mode)

Input data PDO *CAS_XXX_PDOn_IN* (*TCAS_XXX_PDOn_IN* structure):

(XXX - communication channel designation, n - data zone sequence acc. to fig. 2.11.3)

Stat - communication status with participant (8-times bool type)

	X	X	X	X	X	X	COM	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

COM - communication status
0 - communication not established, following data are not valid
1 - communication established, following data valid

Err - not used for now (usint type)

NumR - length of received data (uint type)

Data [x] - received data (usint array element type) - valid only in the master status and for *Cont.EN* = 1

Output data PDO *CAS_XXX_PDOn_OUT* (*TCAS_XXX_PDOn_OUT* structure):

(XXX - communication channel designation, n - data zone sequence acc. to fig. 2.11.3)

Cont - communication control (usint type)

	EN	X	X	X	X	X	X	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

EN - activation of this input and output data zone
0 - this input and output data zone is not used
1 - this input and output data zone is used for communication with the station, the address of which is specified in the variable *Adrs*

Adrs - slave station current address (usint type)

NumT - length of transmitted data (uint type)

Data [x] - transmitted data (usint array element type) - valid only in the master status and for *Cont.EN* = 1

2.11.2.1. Slave mode operation

In the slave mode only one receiving and one transmitting data zone is declared. PDOs of this station are mapped to these zones according to the Table 2.11.1.

Table.2.11.1 Mapping of PDO in the slave mode

PDO object	Symbolic name (XXX - serial channel designation)
RxPDO1	CAS_XXX_PDO_IN~Data[0] to [7]
RxPDO2	CAS_XXX_PDO_IN~Data[8] to [15]
RxPDO3	CAS_XXX_PDO_IN~Data[16] to [23]
RxPDO4	CAS_XXX_PDO_IN~Data[24] to [31]
TxPDO1	CAS_XXX_PDO_OUT~Data[0] to [7]
TxPDO2	CAS_XXX_PDO_OUT~Data[8] to [15]
TxPDO3	CAS_XXX_PDO_OUT~Data[16] to [23]
TxPDO4	CAS_XXX_PDO_OUT~Data[24] to [31]

Concrete symbolic names can be seen in the panel V/V settings (see fig. 2.11.6). Data are arranged from the lowest to the highest byte (Intel convention).

The status of communication can be found in the bit *COM* in the variable *Stat*. If the bit is set to log.1, data exchange takes place.

2.11.2.2. Floating master mode operation

In the floating master mode one receiving and one transmitting data zone is declared for the slave status and also other 1 to 7 receiving and transmitting data zones for the master status.

After the user program restart, the station is in the slave status but it is not operational, since no address is entered. To start station communication, we must set the control bytes to the required status and restart the network using the *REST* bit in the *Control* variable in all data zones.

By setting this bit to log.1 all data in variables *Cont* and *Adrs* in all transmitting data zones will be accepted. Then, the *REST* bit is deleted automatically. Any change to the variables *Cont* and *Adrs* with the zero value of the bit *REST* has no effect.

Any data exchange including control and status ones between the scratchpad and the channel driver takes place exclusively at the I/O scan so that the order of entered or read values is of no importance.

Switching to the slave status

To switch to the slave mode, it is necessary to enter the station address and to set the slave mode.

```
;set slave status in this station
LD   2
WR   CAS_CH2_PDO_OUT~ADRS       ; address of this station is 2
LD   0
WR   CAS_CH2_PDO_OUT~CONT.MAS   ; slave status

; other zones are automatically disconnected
; accept changes
LD   1
WR   r0_p2_Control_CH2~REST      ; restart
```

Data are transmitted within data zones of this station, in the same manner as in the slave mode (chapter 12.11.2.1.). Other zones are off.

The status of communication can be seen in the *COM* bit of the *Stat* variable in the receiving zone of this station.

If the bit is set to log.1, data exchange takes place. The *MAS* bit in the same variable is set to log.0 which signalises the slave status.

Switching to the master status

For switching to the master status, it is necessary to enter station addresses, to activate the corresponding data zones and to set the master status.

```
;set master status in this station
LD   2
WR   CAS_CH2_PDO_OUT~ADRS        ; address of this station is 2
LD   1
WR   CAS_CH2_PDO_OUT~CONT.MAS    ; master status

; data zones PDO1 to be used for communication with station 3
LD   3
WR   CAS_CH2_PDO1_OUT~ADRS       ; address of first slave station
                                 ; is 3
LD   1
WR   CAS_CH2_PDO1_OUT~CONT.EN    ; activate zones

; data zones PDO2 to be used for communication with station 6
LD   6
WR   CAS_CH2_PDO2_OUT~ADRS       ; address of second slave station
                                 ; is 6
LD   1
WR   CAS_CH2_PDO2_OUT~CONT.EN    ; activate zones

; do not use PDO3 data zones
LD   0
WR   CAS_CH2_PDO2_OUT~CONT.EN    ; switch off zones

; accept changes
LD   1
WR   r0_p2_Control_CH2~REST      ; restart
```

Data are transmitted exclusively in data zones of the slave stations. Only status and control bytes are active in data zones of this station.

The *MAS* bit in the *Stat* variable in the receiving zone of this station is set to log.1 which is signalised by the master status.

The status of communication with the relevant station can be seen in the *COM* bit in the *Stat* variable in receiving zones of slave stations. If the bit is set to log.1, data exchange is running.

2.12. CAB MODE - CAN BUS CONNECTION

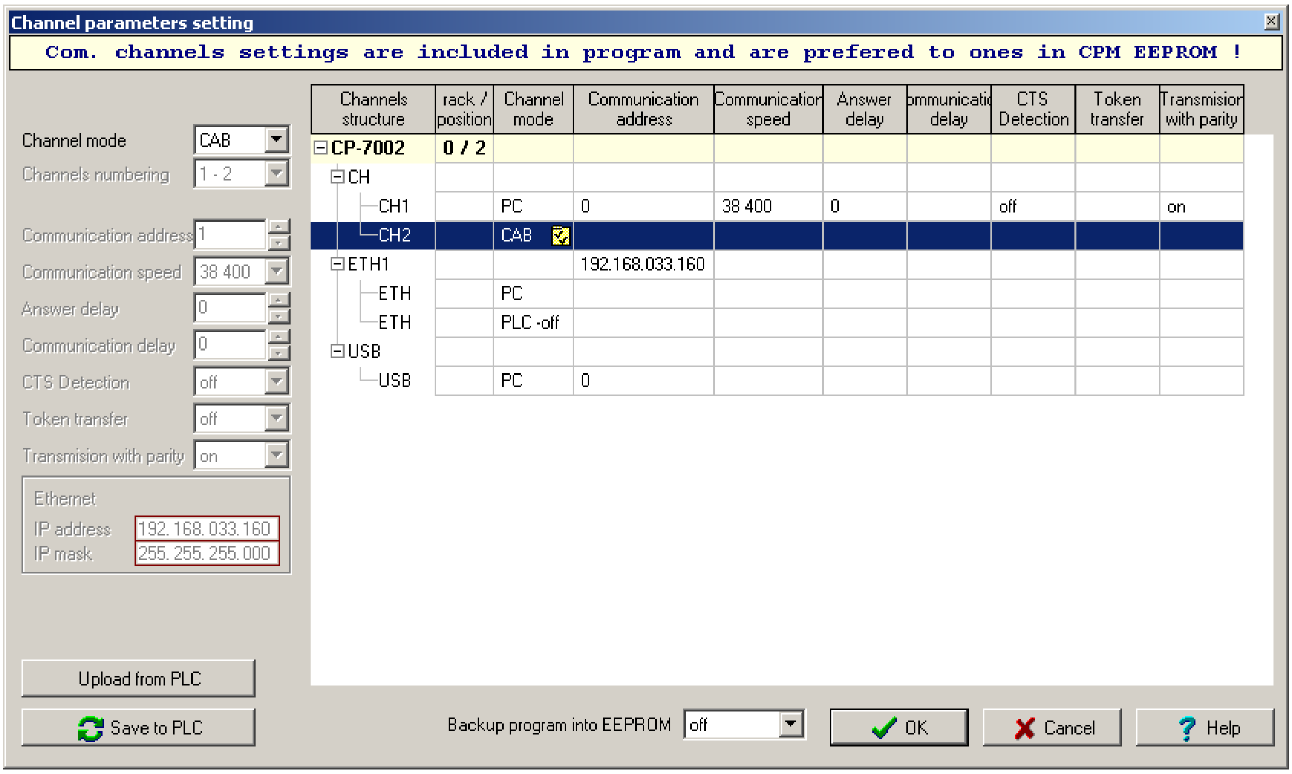
The **CAB** mode allows a general operation of the CAN bus which is connected via the MR-0151 submodule. The transmission rate is optional 500, 250, 125, 50, 20 or 10 kBd.

The **CAB** mode can be set only when the MR-0151 submodule is fitted otherwise the channel is switched off automatically and an error during channel initialisation occurs. When the MR-0151 module is fitted, the **CAN**, **CAS, CAB** or **OFF** mode can be set only and no other modes can be set. The type of submodules can be found in the Mosaic development environment in the option *PLC* | *HW configuration* in the tag *Module info* (see chapter 1.1.).

The MR-0151 submodule is mounted by I82527 sequencer which uses the principle of parallel communication with 15 communication cartridges and messages sorting according to identificators. The advantage is the possibility of preferential message processing with designated identificators. Disadvantages are problems with processing of more messages with the similar identificator within a short time period or with processing of messages with a large number of identificators that can not be sorted effectively. In such a case, it is Berger to use the MR-0161 submodule and the **CSJ** mode (see chapter 2.13.).

2.12.1. Parameters setting

In the Mosaic development environment in the Project manager, we select the folder *Hw* | *HW Configuration*. In the list of modules we highlight, using the mouse, the central unit, the channel of which we want to set. Then we press the push-button *Setting* or the icon ☑ on the line on the right from the position number. The panel *Channel parameters setting* (fig. 2.12.1) will be displayed.



Channel parameters setting

Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

Channel mode: CAB
Channels numbering: 1 - 2
Communication address: 1
Communication speed: 38 400
Answer delay: 0
Communication delay: 0
CTS Detection: off
Token transfer: off
Transmision with parity: on

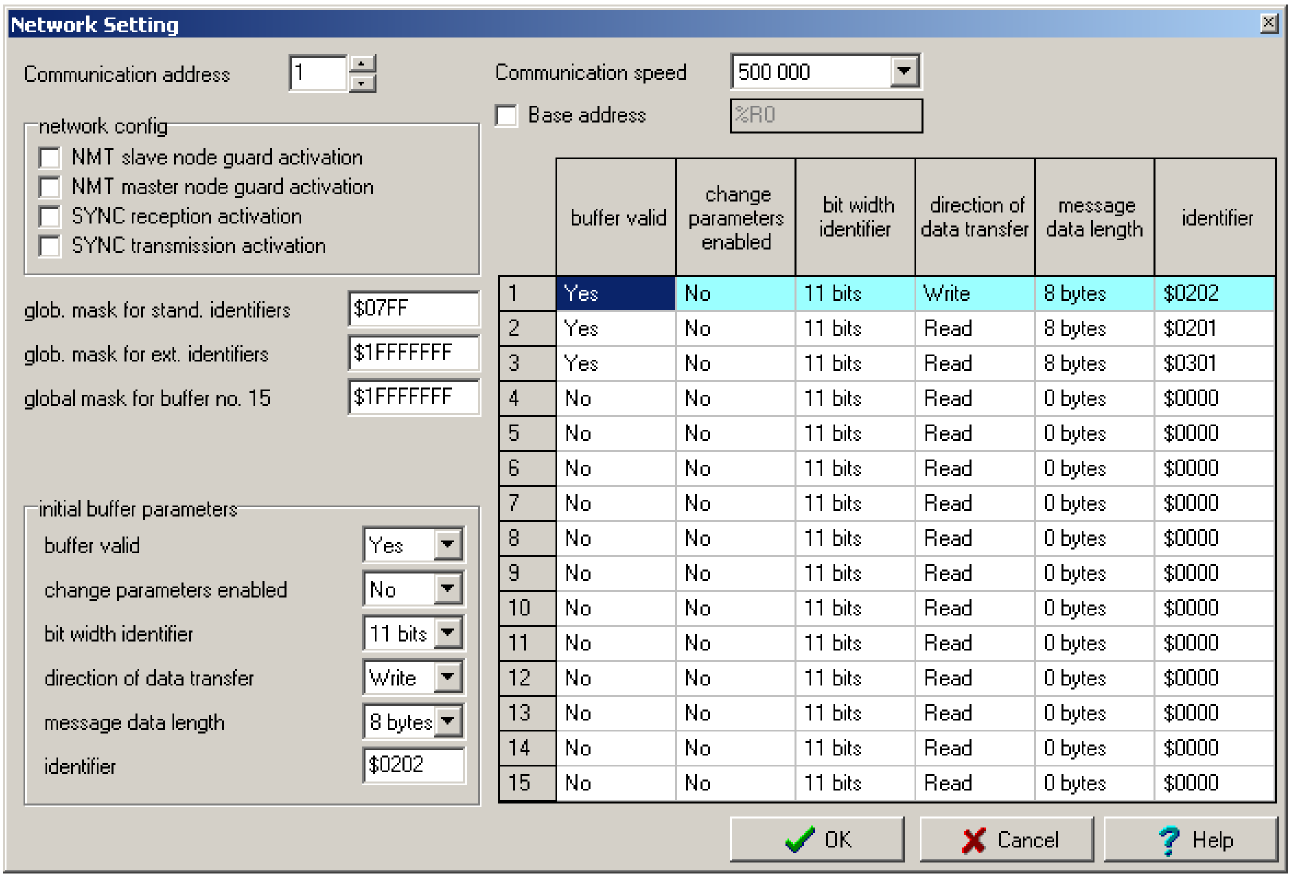
Ethernet
IP address: 192. 168. 033. 160
IP mask: 255. 255. 255. 000

Channels structure	rack / position	Channel mode	Communication address	Communicatior speed	Answer delay	ommunicatio delay	CTS Detection	Token transfer	Transmisior with parity
⊟ CP-7002	0 / 2								
⊟ CH									
CH1		PC	0	38 400	0		off		on
CH2		CAB ☑							
⊟ ETH1			192.168.033.160						
ETH		PC							
ETH		PLC -off							
⊟ USB									
USB		PC	0						

Backup program into EEPROM: off

Fig.2.12.1 ***CAB*** *mode setting*

Set the corresponding serial channel to the **CAB** mode. Then we press the push-button ☑ on this line and the window *Network setting* (fig.2.12.2) show up. Here, parameters for the CAN bus sequencer operation can be set.



Network Setting

Communication address: 1

Communication speed: 500 000

□ Base address: %R0

network config

□ NMT slave node guard activation
□ NMT master node guard activation
□ SYNC reception activation
□ SYNC transmission activation

glob. mask for stand. identifiers: $07FF
glob. mask for ext. identifiers: $1FFFFFFF
global mask for buffer no. 15: $1FFFFFFF

initial buffer parameters

buffer valid: Yes
change parameters enabled: No
bit width identifier: 11 bits
direction of data transfer: Write
message data length: 8 bytes
identifier: $0202

	buffer valid	change parameters enabled	bit width identifier	direction of data transfer	message data length	identifier
1	Yes	No	11 bits	Write	8 bytes	$0202
2	Yes	No	11 bits	Read	8 bytes	$0201
3	Yes	No	11 bits	Read	8 bytes	$0301
4	No	No	11 bits	Read	0 bytes	$0000
5	No	No	11 bits	Read	0 bytes	$0000
6	No	No	11 bits	Read	0 bytes	$0000
7	No	No	11 bits	Read	0 bytes	$0000
8	No	No	11 bits	Read	0 bytes	$0000
9	No	No	11 bits	Read	0 bytes	$0000
10	No	No	11 bits	Read	0 bytes	$0000
11	No	No	11 bits	Read	0 bytes	$0000
12	No	No	11 bits	Read	0 bytes	$0000
13	No	No	11 bits	Read	0 bytes	$0000
14	No	No	11 bits	Read	0 bytes	$0000
15	No	No	11 bits	Read	0 bytes	$0000

Fig.2.12.2 ***CAB*** *mode parameters setting*

Communication address - address of this station (nod). The station address is used in the optional NMT Node Guard function. It must not be identical to any address of any CAN network participant and must not be 0

Communication speed - CAN network transmission rate, all stations within the CAN network must be of the same rate.

Base address - after enabling this option, it is possible, in the following field, to enter the initial data protocol of the communication object 1, data of other objects immediately following

Network config. - the activation of services that are used by CANopen bus (acc. to DS301 norm of CiA association)

NMT slave node guard activation - the station reacts to cyclic questions of the NMT master station concerning its own status (see chapter 2.12.2.2.)

NMT master node guard activation - the station identify the status of other stations in the network (see chapter 2.12.2.2.)

SYNC reception activation - the station receives synchronizational messages (see chapter 2.12.2.3.)

SYNC transmission activation - the station sends synchronizational messages (see chapter 2.12.2.3.)

The CAN bus sequencer I82527 contains 15 communication buffers which enable to operate 15 communication objects simultaneously (COB). To each communication buffer, there can be set the identifier, transmission route and number of data within the range of 0 to 8 bytes. Moreover, there can be allowed the reception of more identifiers to one communication buffer using the mask.

Global mask for standard identifiers - the mask for identifiers of 11 bits length within communication buffers 1 - 14 set for reception

Global mask for extended identifiers - the mask for identifiers of 29 bits length within communication buffers 1 - 14 set for reception

Global mask for buffer no. 15 - the mask for identifier within communication buffer no. 15 set for reception

It is applicable for all masks that they indicate the validity of the identifier in the relevant buffer within those bits where the mask has the value of log.1. Thus, if the identifier in the buffer designated to reception has the value $0340 and the corresponding mask has the value $07FC, then this buffer will receive data with $0340, $0341, $0342 and $0343. The mask for buffers 1 - 14 is common , the buffer 15 has its own mask. It results from this that for purposes of reception of more identifiers, it is the most suitable one. The setup of every of 15 buffers is as follows:

Buffer valid - buffer will be used, set data are valid
Change parameters enabled - identifier and data length will be downloaded straight from the scratchpad, therefore, they can be changed while in operation (cyclic transfers can not be used)
Bit width identifier - selection of the width of the identifier 11 bits or 29 bits
Direction of data transfer - reading or entering
Message data length - 0 to 8 bytes
Identifier - the value of the message identifier

If we activate any of the CANopen function, then this function will occupy some buffers. These buffers are marked with a text - *This buffer is engaged by NMT or SYNC function* - and are unavailable for the user.

The detailed description of the operation is showed in the chapter 2.12.2.1.

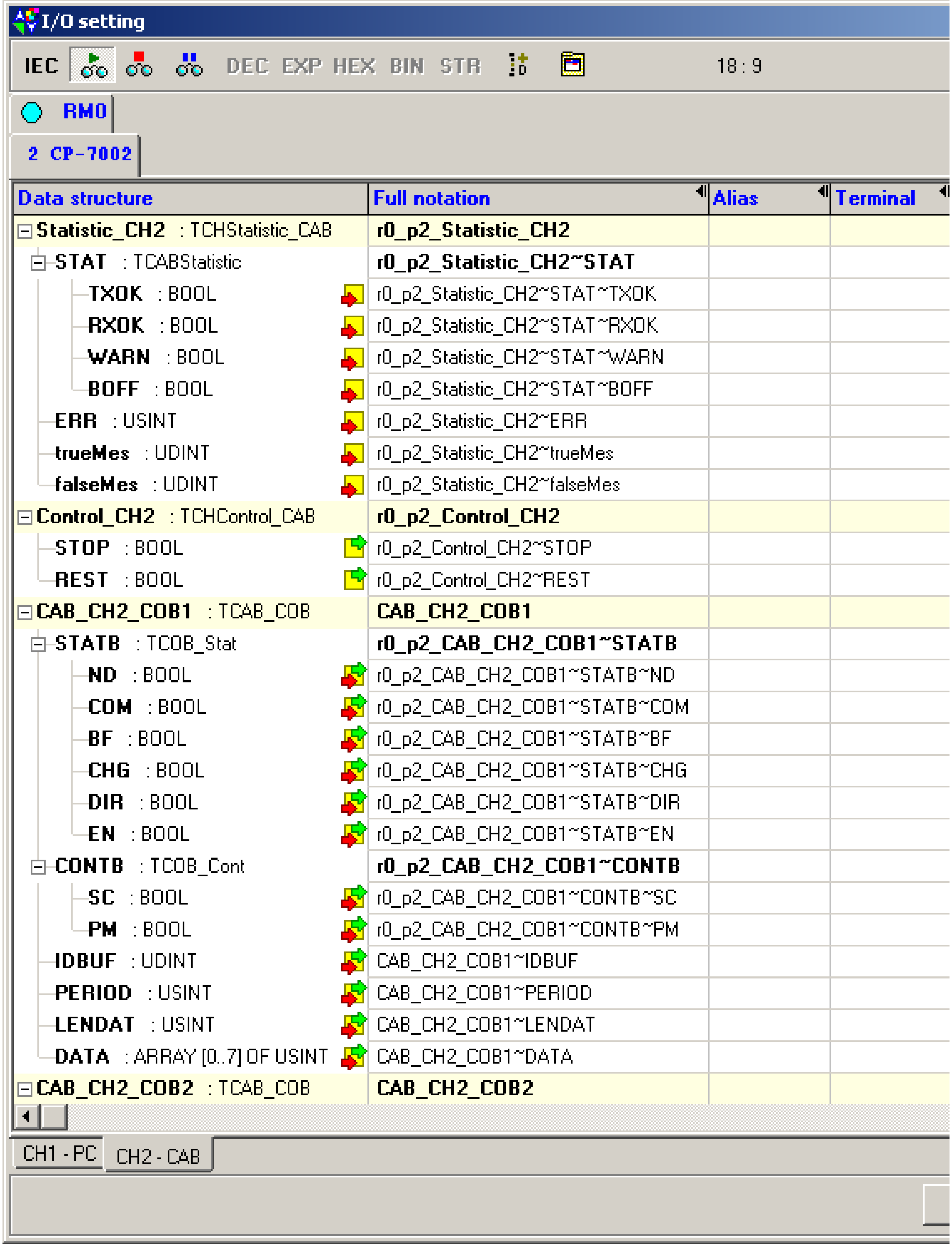
2.12.2. Network operation

The network operation runs on the background of the user program asynchronously to the cycle period of the user program. However, the interchange of current data with the scratchpad proceeds always within the cycle turn of the user program.

Operation and network data diagnostic

The communication channel in the **CAB** mode publish diagnostics data of the CAN sequencer status, received data and sent data. These data are saved in the scratchpad and are easily accessible via the panel *I/O setting* that is accessible through the IO icon (fig. 2.12.3).

Data have assigned symbolic names that begin with the rack and position number. In the column *Full notation,* there is always stated a concrete symbolic name for the set item. If we want to use data in the user program, we ether use this symbolic name or we enter in the column *Alias* our own symbolic name that can then be used. In no case do we use absolute operands as they can alter after the new compilation of the user program.



Data structure	Full notation	Alias	Terminal
Statistic_CH2 : TCHStatistic_CAB	r0_p2_Statistic_CH2		
STAT : TCABStatistic	r0_p2_Statistic_CH2~STAT		
TXOK : BOOL	r0_p2_Statistic_CH2~STAT~TXOK		
RXOK : BOOL	r0_p2_Statistic_CH2~STAT~RXOK		
WARN : BOOL	r0_p2_Statistic_CH2~STAT~WARN		
BOFF : BOOL	r0_p2_Statistic_CH2~STAT~BOFF		
ERR : USINT	r0_p2_Statistic_CH2~ERR		
trueMes : UDINT	r0_p2_Statistic_CH2~trueMes		
falseMes : UDINT	r0_p2_Statistic_CH2~falseMes		
Control_CH2 : TCHControl_CAB	r0_p2_Control_CH2		
STOP : BOOL	r0_p2_Control_CH2~STOP		
REST : BOOL	r0_p2_Control_CH2~REST		
CAB_CH2_COB1 : TCAB_COB	CAB_CH2_COB1		
STATB : TCOB_Stat	r0_p2_CAB_CH2_COB1~STATB		
ND : BOOL	r0_p2_CAB_CH2_COB1~STATB~ND		
COM : BOOL	r0_p2_CAB_CH2_COB1~STATB~COM		
BF : BOOL	r0_p2_CAB_CH2_COB1~STATB~BF		
CHG : BOOL	r0_p2_CAB_CH2_COB1~STATB~CHG		
DIR : BOOL	r0_p2_CAB_CH2_COB1~STATB~DIR		
EN : BOOL	r0_p2_CAB_CH2_COB1~STATB~EN		
CONTB : TCOB_Cont	r0_p2_CAB_CH2_COB1~CONTB		
SC : BOOL	r0_p2_CAB_CH2_COB1~CONTB~SC		
PM : BOOL	r0_p2_CAB_CH2_COB1~CONTB~PM		
IDBUF : UDINT	CAB_CH2_COB1~IDBUF		
PERIOD : USINT	CAB_CH2_COB1~PERIOD		
LENDAT : USINT	CAB_CH2_COB1~LENDAT		
DATA : ARRAY [0..7] OF USINT	CAB_CH2_COB1~DATA		
CAB_CH2_COB2 : TCAB_COB	CAB_CH2_COB2		

Fig.2.12.3 Communication channel data in the ***CAB*** *mode*

Network diagnostics:
Input data *rx_py_Statistic_XXX* (*TCHStatistic_CAB structure*):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Stat - CAN bus status (8-times bool type)

	BOFF	WARN	X	RXOK	TXOK	X	X	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

TXOK - message sending to the CAN bus
0 - valid messages are not sent
1 - valid messages are sent

RXOK - messages reception on the CAN bus
0 - valid messages are not received
1 - valid messages are received

WARN - error occurance on the CAN bus
0 - excessive amount of errors not detected on the CAN bus
1 - excessive amount of errors occurred on the CAN bus, their number exceeded 96

BOFF - the connection of the sequencer to the CAN bus
0 - sequencer is connected to the CAN bus
1 -abnormal error occurance on the CAN bus, their number exceeded 256. The sequencer was disconnected from the bus (bus-off status) and requires a new initialization via REST bit in the variable Control

Err - Communication error notified by the CAN sequencer (usint type)

0 - no error

1 - stuff error
More than 5 similar bits occurred in the sequence in the part of the received message where it is not allowed

2 - form error
The fixed format zone of the received message has a false format.

3 - acknowledgment error
Sent message was not confirmed by any nod.

4 - bit 1 error
During the message sending (excluding arbitrary array), the sequencer sent the bit in log.1 but the status of the bus was log.0.

5 - bit 0 error
During the message sending (excluding arbitrary array), the sequencer sent the bit in log.0 but the status of the bus was log.1. During the bus-off status this means a permanent error.

6 - CRC error
CRC error control character in the received message is false. .

trueMes - number of valid network communication (udint type)
falseMes - number of invalid network communication (udint type)

Output data *rx_py_Control_XXX* (*TCHControl_CAB structure*):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Control - communication control (uint type)

	REST	X	X	X	X	X	X	STOP
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

	X	X	X	X	X	X	X	X
bit	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*

STOP - network control (valid only while NMT master node guard function is active)
0 - station in operation mode
1 - all stations in STOP mode, outputs blocked (providing stations support this status)

REST - CAN controller restart
0 - the network is operated with parameters specified during the last restart
1 - perform CAN controller restart according to new parameters in data zones (it is necessary if the controller is in bus-off state, the bit is automatically deleted)

Communication object data:
***CAB_XXX_COBn* (*TCAB_COB structure*):**

(XXX - communication channel designation, n - communication object number)

Statb - communication status with participant (8-times bool type)

	EN	DIR	CHG	X	X	BF	COM	ND
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

ND - new data reception flag
0 - new data was not received
1 - new data received - the flag needs to be reset by the user program

COM - station-node status (active function NMT master node guard)
0 - station either does not communicate or is not in the operational mode
1 - station is in the operation mode

BF - communication object configuration
0 - com. object ready for other requirements
1 - com. object occupied by the previous requirement

CHG - com. object parameters change permission
0 - com. object parameters can not be changed, identifier and data length are set by initialisation
1 - identifier and data length of the com. object are loaded from variables *IDbuf* and *Lendat* and thus can be changed

DIR - com. object transmission route
0 -reception
1 - sending

EN - com. object enabled
0 - com. object switched off
1 - com. object switched on

Contb - communication control (usint type)

	X	X	X	X	X	X	PM	SC
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

SC - new data transfer request
0 - data not transferred
1 - data transfer request
If *PM* = 0, then it is a single transfer and *SC* bit is reset automatically after the transfer is undertaken. If PM = 1, then it is a cyclic transfer that runs continuously until we reset the SC bit using the user program. The cycle period is conditioned by the variable *Period* consent*.*

PM - data transfer type
0 - single data transfer
1 - cyclic data transfer

IDbuf - com. object identifier (udint type)
If there is during the initialisation of this com. object set the standard identifier, then its value lies within the lowest 11 bites of the variable. If there is the extended identifier set, then its value lies within the lowest 29 bites of the variable. If the data route is set for broadcasting, then immediately after the initialisation, there is into the variable *IDbuf* entered the value of the identifier that was set within the initialisation.
During the data reception, there is into the variable *IDbuf* entered the actual value of the incoming identifier.
If the change of the com. object is allowed, then during the request on the data entry or data reading there is used as an identifier the consent of the variable *IDbuf*.

Period - cyclic transfer repeating period in 10 ms (usint type)

Lendat - transferred data length (uint type)
If the data transfer is set for the sending, then immediately after the initialisation there is into the variable *Lendat* entered the number of transferred data that was set within the initialisation.
During the data reception there is into the variable *Lendat* entered the actual length of data received.
If the change of the com. object is allowed, then during the request on the data entry there is as an actual data length used the consent of the variable *Lendat*.

Data [x] - data transferred (usint array type)

2.12.2.1. Data interchange

The reception of data sent by another node

If we want to receive data broadcasted from another node (station), then we set within the initialisation of the communication object the corresponding identifier, data length and reception transfer route. Consequently, data containing the corresponding identifier are received at the moment when other node send them. The data reception is indicated by the *Statb.ND* bit that is set to log. 1 during the next nearest cycle scan. We set it back to log. 0 using the user program.

If we want to receive data containing various identifiers in one communication object, we achieve this by setting masks of communication objects. (see chapter 2.12.1.). The most suitable is to use the communication object 15 that has a separated mask from other objects.

Data loading from another node

If we want to read data from another node (station) that does not actively send these data, we set within the initialisation of the com. object the corresponding identifier, data length and reception transfer route.

In the user program we set *Contb.SC* bit into the log.1 to the corresponding com. object whereby we send a single request on data. The bit is automatically reset. Data containing the relevant identifier are received subsequently which is indicated by *Statb.ND* bit that is set to log.1 in the next nearest cycle scan. We set it back to log.0 using the user program.

If we want to read data cyclically, we enter into the variable *Period* a value in the range of 1 - 255 (i.e. 10 - 2550 ms), then we set *Contb.PM* bit to log.1 (cyclic communication) and *Contb.SC* bit to log.1 (request on data). Data will be loaded with the set period until we set the *Contb.SC* bit to log.0.

If we want to read data with various identifiers within one com. object, we enable during the initialisation the change of parameters of the com. object (see chapter 2.12.1.). Then, during each single request on data, values of the identifier and data length set in variables *IDbuf* and *Lendat* are used. Cyclic communication is not enabled in this case.

Let’s draw the attention to the *Statb.BF* bit that indicates the com. object occupation by previous request. As long as the *Statb.BF* bit = log1, should we not set another requirement.

Data sending to another node

If we want to send data to another node (station), we set within the initialisation of the com. object the corresponding identifier, data length and sending transfer route.

In the user program we set *Contb.SC* bit to log.1 to the com. object whereby we send the single request on data sending. The bit is automatically reset. Data with the corresponding identifier are sent subsequently.

If we want to send data cyclically, we enter in the variable *Period* the value in the range of 1 - 255 (i.e. 10 - 2550 ms), then we set *Contb.PM* bit to log.1. (cyclic communication) and *Contb.SC* bit to log.1 (request on data sending). Data will be sent with the set period until we set *Contb.SC* bit to log.0.

If we want to send data containing various identifiers within one communication object, we enable the change of parameters of the com. object during the initialisation (see chapter 2.12.1.). Then, during each single request on data sending, values of the identifier and data length set in *IDbuf* and *Lendat* variables, are used. Cyclic communication is not enabled in this case.

Let’s draw the attention to the *Statb.BF* bit which indicates the com. object occupation by the previous requirement. As long as the *Statb.BF* bit = log1, should we not set another requirement.

2.12.2.2. Function NMT node guard

Function NMT node guard is defined for the CANopen bus by the DS301 norm of the CiA association. It consists in periodical requests of the NMT master station on NMT slave stations (nodes) asking in which status they are in. Asked stations answer whether they are or are not in the operation status. If the NMT slave station is in the STOP status, the NMT master station will try to set it to the operation status.

NMT master

If we tick within the initialisation the *Function NMT master node guard* selection, then this station is NMT master station and undertakes cyclic requests on other stations that must in this case have the NMT slave node guard function active. The list of station (nodes) addresses compiled by master from identifiers set to individual com. objects during the initialisation. It applies that the lowest 7 identifiers assess the station (node) address. In the list, there is opt out the home address that was set within the initialisation and furthermore, identifiers of com. objects that have the change of parameters enabled are not included.

Considering the above mentioned, it is necessary to select values of identifiers that correspond with addresses of relevant stations (nodes).

The status of relevant station is indicated by *Statb.COM* bit in the corresponding com. object. If the bit has the value of log.1, the station is in the operation mode and is able to supply data. . If the bit has the value of log.0, the station is ether in the STOP mode and do not supply data or does not communicate in any way.

Using the *Control.STOP* bit within the diagnostic network zone, we can assign all station in the network to the STOP status (log.1) and back to the operation status (log.0) providing they support this function.

Function NMT master node guard engage com. objects 12 and 13. These are then not accessible to the user.

NMT slave

If we tick within the initialisation the *Function NMT slave node guard* selection then this station is NMT slave and react to the cyclic requests of the NMT master station. The address set during the initialisation is considered as the address of the station (node).

PLC TECOMAT does not react to the notice requiring the transfer to the STOP status that was send by the master station. While the user program is running (RUN mode), the operation status is permanently indicated. If the user program is stopped from any reason (HALT mode), the STOP status is indicated.

Function NMT slave node guard engages com. objects 10 and 11. These are then not accessible to the user.

2.12.2.3. Function SYNC

Function SYNC is defined for the CANopen bus by the DS301 norm of the CiA association. It consists in periodical sending of the SYNC message of the designated station that is received by all other stations. Thus, the network is synchronised.

If we tick the *Provider of the SYNC function,* then this station send SYNC messages. If we tick within the initialisation the *Recipient of the SYNC function,* then this station receive messages SYNC and synchronize according to them the data exchange in communication objects of the CAN sequencer (this does not apply to the PLC scratchpad - here, data are changed at the user program scan only).

Function SYNC engages the com. object 14 which is then not accessible to the user.

2.13. CSJ MODE - CAN BUS CONNECTION

CSJ mode function

CSJ mode allows the general operation of the CAN bus connected via MR-0160 or MR-0161 submodule. The transmission rate is optional 1000, 500, 250, 125, 50 or 20 kBd.

CSJ mode can be set only when the submodule MR-0160 or MR-0161 is mounted, otherwise the channel is switched off automatically and an error occurs during the channel initialisation. As well as the mounted submodule MR-0160 or MR-0161, it is not possible to set other channel modes than **CSJ** or **OFF**. The mounting of submodules can be found in the Mosaic environment via the *PLC* **|** *HW configuration* on the tag *Module info* (see chapter 1.1.).

The difference to the **CAB** mode is mainly that the **CSJ** mode uses submodules mounted with SJA1000 sequencer that receives messages from the CAN bus to the Stack of FIFO type, i. e. sequentially in such order that they were received and without sorting them according to identifiers. The advantage of this organisation approves itself especially in a case when more messages with the same identifier are processed within a short time period or in such a case when messages with a high number of identifiers are processed that can not be organised effectively.

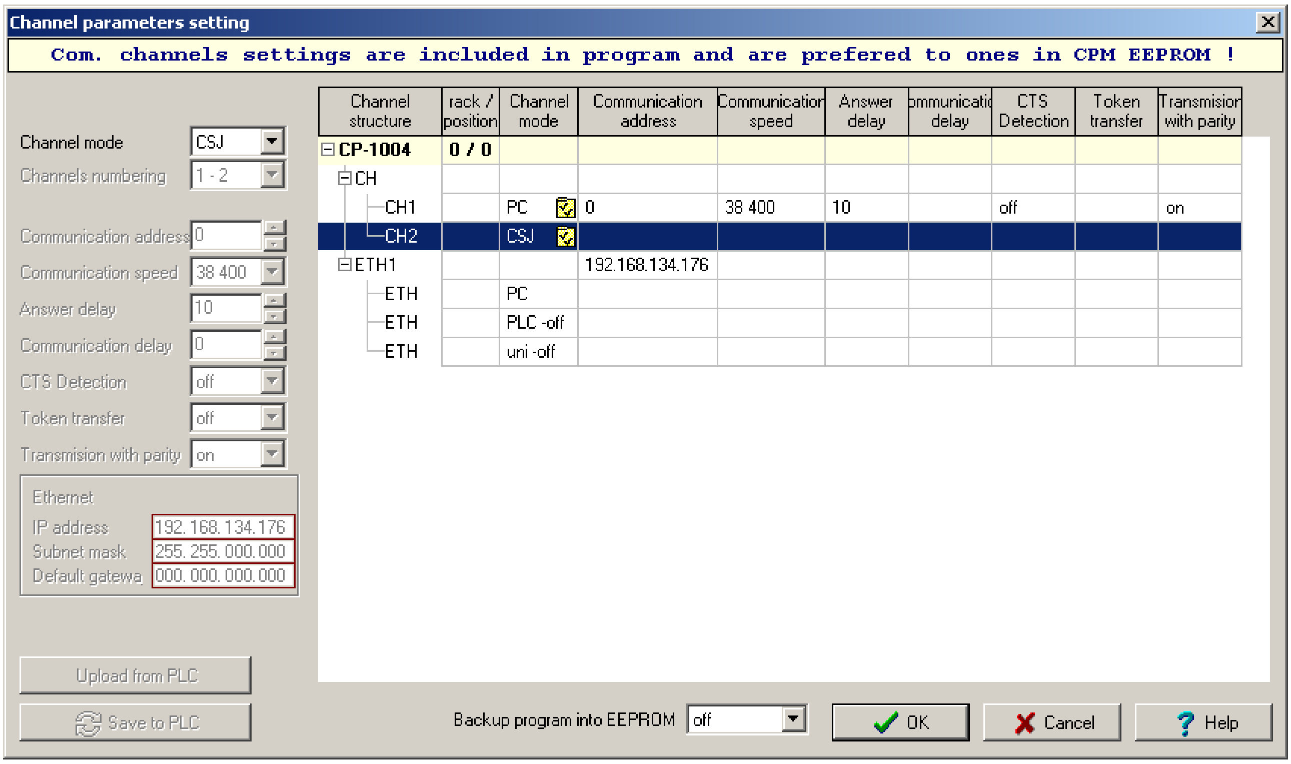
One or two CAN buses within one communication channel

The MR-0161 submodule contains one SJA1000 sequencer while MR-0160 submodule contains two of these sequencers so it enables to operate two independent CAN buses simultaneously. Both buses are in this case operated within one communication channel in the **CSJ** mode.

2.13.1. Parameters setup

In the Mosaic environment in the project manager we select the *Hw* **|** *Configuration HW* item. In the list of individual modules we select, using the mouse, the central unit, the channel of which we want to set. Then we press the push-button *Setting* or the ☑ icon on the line on the right from the position number. The panel *Channel parameters setting* will appear (fig.2.13.1).

We set the corresponding serial channel to the **CSJ** mode. Then we press ☑ icon on this line and the window *Network setting* will pop up (fig.2.13.2). Here we set parameters for CAN bus sequencers operating.



Channel parameters setting

Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

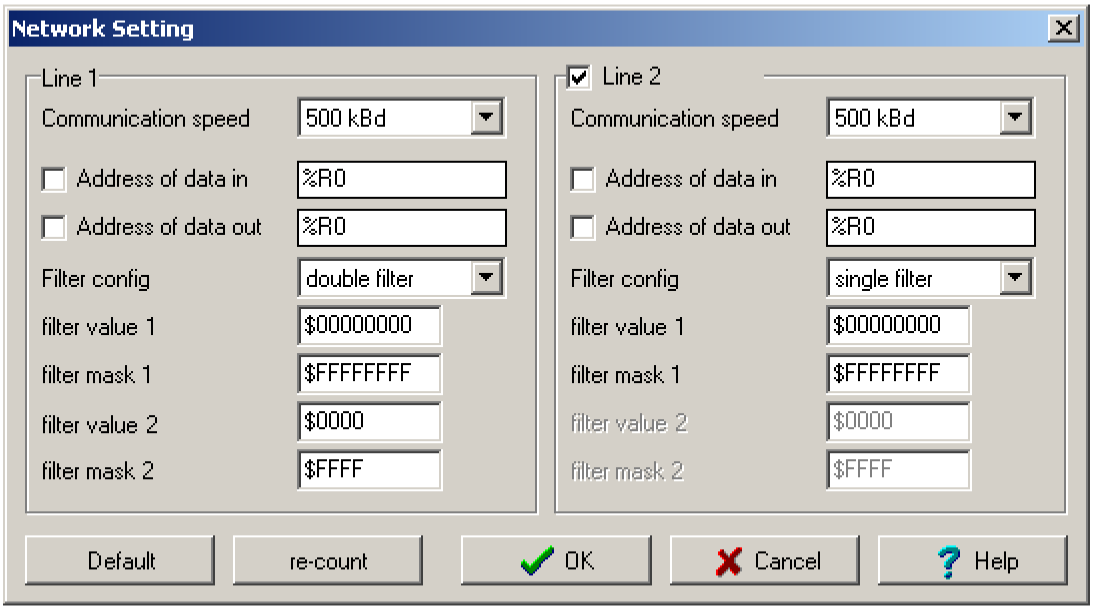
Channel mode: CSJ
Channels numbering: 1 - 2
Communication address: 0
Communication speed: 38 400
Answer delay: 10
Communication delay: 0
CTS Detection: off
Token transfer: off
Transmision with parity: on

Ethernet
IP address: 192. 168. 134. 176
Subnet mask: 255. 255. 000. 000
Default gatewa: 000. 000. 000. 000

Channel structure	rack / position	Channel mode	Communication address	Communicatior speed	Answer delay	ommunicatic delay	CTS Detection	Token transfer	Transmisior with parity
⊟ **CP-1004**	**0 / 0**								
⊟ CH									
CH1		PC ☑	0	38 400	10		off		on
CH2		CSJ ☑							
⊟ ETH1			192.168.134.176						
ETH		PC							
ETH		PLC -off							
ETH		uni -off							

Backup program into EEPROM: off

Fig.2.13.1 ***CSJ*** *mode setting*



Network Setting

Line 1
Communication speed: 500 kBd
□ Address of data in: %R0
□ Address of data out: %R0
Filter config: double filter
filter value 1: $00000000
filter mask 1: $FFFFFFFF
filter value 2: $0000
filter mask 2: $FFFF

☑ Line 2
Communication speed: 500 kBd
□ Address of data in: %R0
□ Address of data out: %R0
Filter config: single filter
filter value 1: $00000000
filter mask 1: $FFFFFFFF
filter value 2: $0000
filter mask 2: $FFFF

Fig.2.13.2 ***CSJ*** *mode parameters setting*

If we use the MR-0160 submodule with two CAN sequencers, the parameters setup of the first CAN network is on the left side of the panel. The setup of the second CAN network is on the right side of the panel. We must tick the *Line 2* box near the heading. If we use the MR-0161 submodule with one CAN sequencer, the parameters setup on the left side of the panel is valid only. The setup of the second network is ignored (it is recommended to switch off the network - the box next to the headline *Line 2* should be not ticked)

Communication speed - the transmission rate of the CAN network, all stations within the CAN network must have this rate on the same level

Address of data in - after ticking this option, it is possible to set in the following field the required initiatory register of the receiving zone, otherwise, it is set by the compiler

Address of data out - after ticking this option, it is possible to set in the following field the required initiatory of the sending zone, otherwise, it is set by the compiler

Filter config. - receiving filter mode
- *single filter*
  one filter testing the first four bytes of the received message incl. identifier
- *double filter*
  two filters testing the first three, or two, bytes of the received message incl. identifier

Filter value 1 - the value that the first filter compare with the beginning of the received message

Filter mask 1 - the mask determining which bits of the first filter value are compared and which are ignored

Filter value 2 - the value that the second filter compare with the beginning of the received message

Filter mask 2 - the mask determining which bits of the second filter value are compared and which are ignored

Using the push-button *Default,* the default parameters setting will be invoked. The receiving filter is then set so, that it lets through all received messages.

Using the push-button *Re-count,* the initial parameters of receiving and sending zones conversion will be invoked in such a way so they lie immediately one after each other and do not overlap. The original initial register that do not change is the initial register of the receiving zone of the line 1. Succeeded by the sending zone of the line 1, receiving zone of the line 2 and sending zone of the line 2 (line 2 only when it is ticked)

Single receiving filter

In this mode, it is possible to define one long filter (4 bytes). The meaning of the relevant filter bits (filter values and masks) and of the beginning of the received message depends on just at the moment receiving message format (the width of the identifier).

Single filter - standard message frame

If the message with the standard frame is received, i. e. the identifier has the width of 11 bits, the whole identifier, the data request flag and first two data bytes are tested by the input filter. If the message has no data or has only one byte of data, then the corresponding number of data is tested.

<table>
<tr><td colspan="8">byte 1</td><td colspan="8">byte 2</td><td colspan="8">byte 3</td><td colspan="8">byte 4</td></tr>
<tr><td colspan="11">ID</td><td>RTR</td><td colspan="4">0</td><td colspan="8">DATA1</td><td colspan="8">DATA2</td></tr>
<tr><td>.7</td><td>.6</td><td>.5</td><td>.4</td><td>.3</td><td>.2</td><td>.1</td><td>.0</td><td>.7</td><td>.6</td><td>.5</td><td>.4</td><td>.3</td><td>.2</td><td>.1</td><td>.0</td><td>.7</td><td>.6</td><td>.5</td><td>.4</td><td>.3</td><td>.2</td><td>.1</td><td>.0</td><td>.7</td><td>.6</td><td>.5</td><td>.4</td><td>.3</td><td>.2</td><td>.1</td><td>.0</td></tr>
</table>

Fig.2.13.3 The structure of the received message with the standard frame for the single filter

ID – message identifier
RTR – data request flag
DATA1 – first data byte
DATA2 – second data byte

All bits at the beginning of the received message must be of the same value as the filter value is, to ensure the valid reception. If we do not want to test some of the bits, we set the filter mask onto the position of these bits to log.1. The lowest 4 bits of the second byte (fig. 2.13.3 marked 0) are not tested. For reasons of compatibility with future versions, it is recommended by the producer of the sequencer to set corresponding bits of the filter mask always to log.1 as not used.

If we want to receive all messages from the CAN bus, we set the filter mask to $FFFFFFFF.

If we want to receive only messages with a concrete identifier, for example $0340, we set the filter mask to $001FFFFF and the filter value to $68000000.

If we want to receive messages with identifiers e. g. $0340, $0341, $0342 and $0343, we set the filter mask to $007FFFFF and filter value to $68000000.

Single filter - extended message frame

If the message with the extended frame is received, i.e. the identifier has the width of 29bits, the whole identifier and data request flag are tested by the input filter.

byte 1	byte 2	byte 3	byte 4		
ID				RTR	0
.7 .6 .5 .4 .3 .2 .1 .0	.7 .6 .5 .4 .3 .2 .1 .0	.7 .6 .5 .4 .3 .2 .1 .0	.7 .6 .5 .4 .3	.2	.1 .0

Fig.2.13.4 The structure of the received message with the extended frame for the single filter
ID – message identifier
RTR – data request flag

To ensure the valid reception, all bits of the beginning of the received message must be of the same value as the filter value is. If we do not want to test some bits, we set the filter mask on the position of these bits to log.1. The lowest 2bits of the fourth byte (fig. 2.13.4 marked 0) are not tested. For reasons of compatibility with future versions, it is recommended by the producer of the sequencer to set corresponding bits of the filter mask always to log.1 as not used.

If we want to receive all messages from the CAN bus, we set the filter mask to $FFFFFFFF.

If we want to receive only messages with a concrete identifier, for example $10000210, we set the filter mask to $00000007 and the filter value to $80001080.

If we want to receive messages with identifiers e. g. $10000210, $10000211, $10000212 and $10000213, we set the filter mask to $0000001F and filter value to $80001080.

Double receiving filter

In this mode, it is possible to define two short filters. Their size and meaning of the correspondent filter bits (filter value and mask) and the beginning of the received message is dependent right on the format of the receiving message (width of the identifier).

Double filter - standard message frame

If the message with the standard frame is received, i. e. the identifier has the width of 11 bits, the whole identifier, data request flag and the first two data bytes are tested by the input filter. If the message contains no data or has only one data byte, the relevant number of data is tested only.

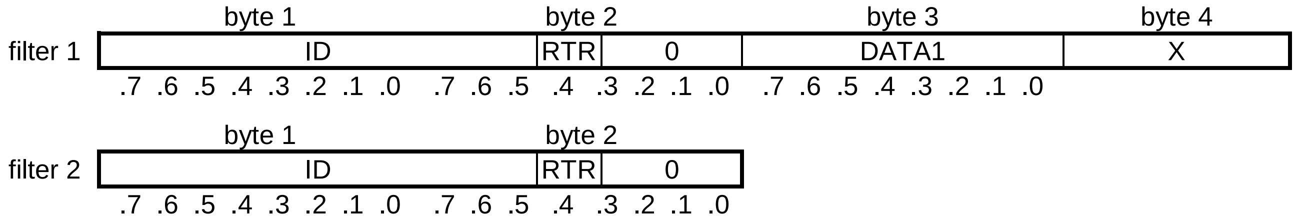


Fig.2.13.5 The structure of the beginning of the received message with the standard frame for the double
ID - message identifier
RTR - data request flag
DATA1 - first data byte
X - not used

To ensure the valid reception, all bits of the beginning of the received message must be of the same value as at least one of the filters is. If we do not want to test some bits, we set the filter mask on the position of these bits to log.1. The lowest 4 bits of the second byte (fig.2.13.5 marked 0) are not tested. For reasons of compatibility with future versions, it is recommended by the producer of the sequencer to set corresponding bits of the filter mask always to log.1 as not used. The fourth byte of the first filter is not used in this mode.

If we want to receive all messages from the CAN bus, we set the first filter mask to $FFFFFFFF and the second filter mask to $FFFF.

If we want to receive only messages with two concrete identifiers, for example $0340 and $0412, we set the first filter mask to $001FFFFF and value to $68000000, the second filter mask to $001F and value to $8240.

Double filter - extended message frame

If the message with the extended frame is received, i.e. the identifier has the width of 29bits, the whole identifier and data request flag are tested by the input filter.

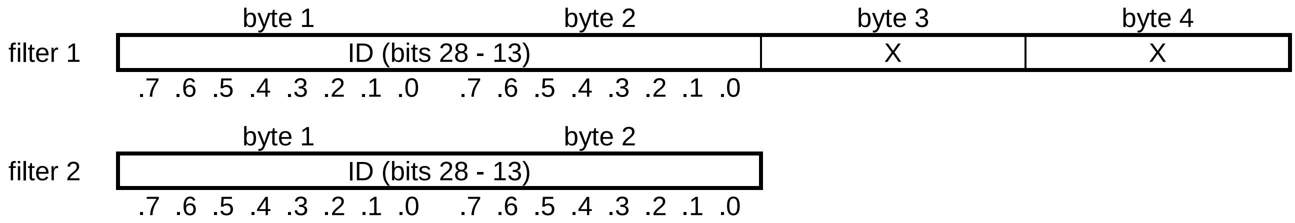


Fig.2.13.6 The structure of the beginning of the received message with the extended frame for the double filter
ID – message identifier (the highest 16 bits)
X – not used

To ensure the valid reception, all bits of the beginning of the received message must be of the same value as at least one of the filters is. If we do not want to test some bits, we set the filter mask on the position of these bits to log.1. The third and fourth bit of the first filter are used in this mode.

If we want to receive all messages from the CAN bus, we set the first filter mask to $FFFFFFFF and the second filter mask to $FFFF.

If we want to receive only messages with two groups of identifiers, for example $10002000 - $10003FFF and $10028000 - $10029FFF, we set the first filter mask to $0000FFFF and value to $80010000, the second filter mask to $0000 and value to $8014.

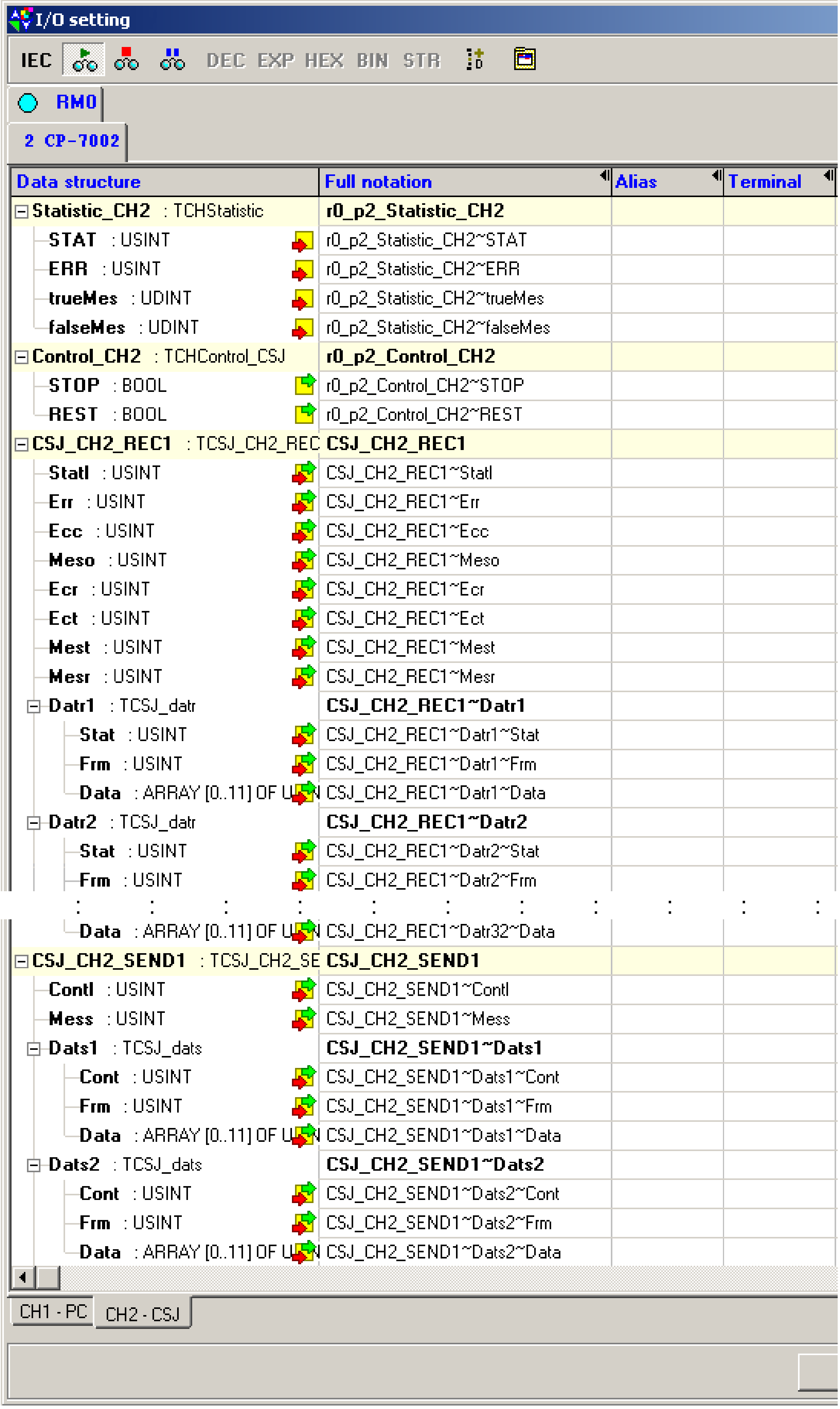
2.13.2. Network operation

The network operation runs on the background of the user program asynchronously with the cycle period of the user program. Nevertheless, the current data exchange with the scratchpad runs always during the cycle scan of the user program.

Operation diagnostic and network data

The communication channel in the **CSJ** mode publish diagnostics data of the CAN sequencer status, received data and sent data. These data are saved to the scratchpad and are easily accessible via the *I/O setting* panel, using the IO icon (fig. 2.13.7).

Data have assigned symbolic names that begin with the rack number and position number. In the column *Full notation,* there is always the concrete symbolic name shown for the given item. If we want to use data in the user program, we use either this symbolic name or in the column *Alias* we enter our own symbolic name that can be then used. In no case do we use absolute operands as they can alter after a new compilation of the user program.



Data structure	Full notation	Alias	Terminal
⊟Statistic_CH2 : TCHStatistic	r0_p2_Statistic_CH2		
STAT : USINT	r0_p2_Statistic_CH2~STAT		
ERR : USINT	r0_p2_Statistic_CH2~ERR		
trueMes : UDINT	r0_p2_Statistic_CH2~trueMes		
falseMes : UDINT	r0_p2_Statistic_CH2~falseMes		
⊟Control_CH2 : TCHControl_CSJ	r0_p2_Control_CH2		
STOP : BOOL	r0_p2_Control_CH2~STOP		
REST : BOOL	r0_p2_Control_CH2~REST		
⊟CSJ_CH2_REC1 : TCSJ_CH2_REC	CSJ_CH2_REC1		
Statl : USINT	CSJ_CH2_REC1~Statl		
Err : USINT	CSJ_CH2_REC1~Err		
Ecc : USINT	CSJ_CH2_REC1~Ecc		
Meso : USINT	CSJ_CH2_REC1~Meso		
Ecr : USINT	CSJ_CH2_REC1~Ecr		
Ect : USINT	CSJ_CH2_REC1~Ect		
Mest : USINT	CSJ_CH2_REC1~Mest		
Mesr : USINT	CSJ_CH2_REC1~Mesr		
⊟Datr1 : TCSJ_datr	CSJ_CH2_REC1~Datr1		
Stat : USINT	CSJ_CH2_REC1~Datr1~Stat		
Frm : USINT	CSJ_CH2_REC1~Datr1~Frm		
Data : ARRAY [0..11] OF U	CSJ_CH2_REC1~Datr1~Data		
⊟Datr2 : TCSJ_datr	CSJ_CH2_REC1~Datr2		
Stat : USINT	CSJ_CH2_REC1~Datr2~Stat		
Frm : USINT	CSJ_CH2_REC1~Datr2~Frm		
: : : :	: : : :	: :	: :
Data : ARRAY [0..11] OF U	CSJ_CH2_REC1~Datr32~Data		
⊟CSJ_CH2_SEND1 : TCSJ_CH2_SE	CSJ_CH2_SEND1		
Contl : USINT	CSJ_CH2_SEND1~Contl		
Mess : USINT	CSJ_CH2_SEND1~Mess		
⊟Dats1 : TCSJ_dats	CSJ_CH2_SEND1~Dats1		
Cont : USINT	CSJ_CH2_SEND1~Dats1~Cont		
Frm : USINT	CSJ_CH2_SEND1~Dats1~Frm		
Data : ARRAY [0..11] OF U	CSJ_CH2_SEND1~Dats1~Data		
⊟Dats2 : TCSJ_dats	CSJ_CH2_SEND1~Dats2		
Cont : USINT	CSJ_CH2_SEND1~Dats2~Cont		
Frm : USINT	CSJ_CH2_SEND1~Dats2~Frm		
Data : ARRAY [0..11] OF U	CSJ_CH2_SEND1~Dats2~Data		

*Fig.2.13.7 Communication channel data in the **CSJ** mode*

Network diagnostics:
Input data *rx_py_Statistic_XXX* (*TCHStatistic_CSJ structure*):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Stat - not used

Err - not used

trueMes - number of valid received messages from both CAN buses (udint type)

falseMes - number of errors of both CAN buses (udint type)

Output data *rx_py_Control_XXX* (*TCHControl_CSJ structure*):

(x - rack number, y - module position number in the rack, XXX - communication channel designation)

Control - not used

Data of the receiving stack of the n bus:
***CSJ_XXX_RECn* (*TCSJ_XXX_RECn structure*):**

(XXX - communication channel designation, n - number of CAN bus)

Statl - sequencer status (8-times bool type)

	RESM	STT	STOP	X	X	X	X	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

STOP - conversion to the reset mode
0 - sequencer status is stable
1 - sequencer converts to reset status
STT - conversion from the reset mode
0 - sequencer status is stable
1 - sequencer converts from the reset status to the operation mode
RESM - sequencer mode
0 - sequencer is in the operation mode
1 - sequencer is in the reset mode

Err - error notified by the sequencer (8-times bool type)

	BS	EW	X	X	X	BE	DOS	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

DOS - size error of the receiving stack in the sequencer
0 - receiving stack of the sequencer runs normally
1 - receiving stack of the sequencer is overloaded
The sequencer contains internal equalizing stack for data reception. If the controller of the communication channel is not able to collect received messages, the stack is overloaded.
BE - bus error
0 - operation within the CAN bus is free and clear
1 - within the CAN bus an error occurred (according to the specification of the CAN 2.0B protocol), similar data on the error is saved in the *Ecc* variable

EW - increased number of errors

0 - number of errors within the bus did not overpass the warning zone

1 - warning due to number of errors, one of the counters *Ecr* or *Ect* reached the value of 96 or even more than this

BS - bus status

0 - CAN bus free and clear

1 - bus-off status

The bus was disconnected due to high number of errors during sending, counter *Ect* reached the value of 255. the sequencer automatically converts to the reset mode. The communication channel controller will attempt to restart it and re-connect it to the bus again.

Ecc - bus error (8-times bool type)

	EC1	EC0	DIR	SEG4	SEG3	SEG2	SEG1	SEG0
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

EC1 - EC0 - error type

00 - bit error

Bit sending to the bus failed (there is a different level on the bus than there is in the bit).

01 - form error

Bit has a different value than it should have according to the protocol.

10 - stuff error

More than 5 successive bits of the same level were received which is forbidden inside the message.

11 - other errors

Other errors within the protocol.

DIR - transmission route

0 - error occurred during sending

1 - error occurred during reception

SEG4 - SEG0 - message segment where the error occurred

00010 - bits 28 to 21 of extended, or bits 10 to 3 of standard identifier

00011 - beginning of the message frame

00100 - bit SRTR

00101 - bit IDE

00110 - bits 20 to 18 of extended, or bits 2 to 0 of standard identifier

00111 - bits 17 to 13 of extended identifier

01000 - CRC sequence

01001 - reserve bit 0

01010 - data array

01011 - data length

01100 - bit RTR

01101 - reserve bit 1

01110 - bits 4 to 0 of extended identifier

01111 - bits 12 to 5 of extended identifier

10001 - active error flag

10010 - interval between messages

10011 - dominant bits toleration

10110 - passive error flag
10111 - error character
11000 - CRC character
11001 - acknowledgement slot
11010 - frame end
11011 - acknowledgement character
11100 - size error flag

Meso - number of lost messages (udint type)
The number of messages that were not received due to the size error of the internal stack of the communication channel

Ecr - counter of received error messages (usint type)
The number of errors detected by the sequencer during the reception. If this counter reach or overpass the value of 96, then in the variable *Err* the EW bit is set. The reception of the faultless message lower the value of the counter by 1.

Ect - counter of sent error messages (usint type)
The number of errors detected by the sequencer during the sending. If this counter reach or overpass the value of 96, then in the variable *Err* the EW bit is set. The sending of the faultless message lower the value of the counter by 1.

Mest - number of non-sent messages (usint type)
The number of messages that have not been sent yet. The counter does not include new requirements from the previous cycle of the user program.

Mesr - number of received messages (usint type)
The number of messages that were received during the last cycle of the user program and are saved in the following fields in the order as they were received.

Datr1 - received message 1 (message structure TCSJ_DATR see chapter 2.13.2.1.)

:

Datr32 - received message 32 (message structure TCSJ_DATR see chapter 2.13.2.1.)

Data of the sending stack of the n bus:
***CSJ_XXX_SENDn* (*TCSJ_XXX_SENDn structure*):**

(XXX - communication channel designation, n - number of CAN bus)

Contl - sequencer control (8-times boolean type)

	RESM	X	X	X	X	X	X	X
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

RESM - sequencer mode
0 - operational mode
1 - reset mode
While changing the bit RESM from log.1 to log.0, there will be the sequencer re-initialisated according to the initialisation table (same as during the restart of the user program).

Mess - number of messages in the sending stack (usint type)
The number of messages that are entered into the sending stack.

Dats1 - message 1 to be sent (message structure TCSJ_DATS see chapter 2.13.2.1.)

:

Dats32 - message 32 to be sent (message structure TCSJ_DATS see chapter 2.13.2.1.)

2.13.2.1. Message structure

Received messages

Items *Datr1* to *Datr32* containing received messages each have the following structure TCSJ_DATR:

Stat - message status - not used (8-times bool type)

Frm - message format (8-times bool type)

	FF	RTR	X	X	DLC3	DLC2	DLC1	DLC0
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

DLC3 - DLC0 - data length
Length of data carried in the message in the range of 0 to 8.
RTR - data request
0 - common data message
1 - one of the nodes of the CAN bus requires data sending with the identifier carried within the message
FF - message frame format
0 - standard frame (identifier of the width of 11 bits)
1 - extended frame (identifier of the width of 29 bits)

Data [x] - message identifier and data (usint item array type)
The structure depends on the frame format
Standard frame:

byte 0	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
byte 1	ID2	ID1	ID0	RTR	0	0	0	0
byte 2	DATA1							
:	:							
byte 9	DATA8							
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

Extended frame:

byte 0	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
byte 1	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
byte 2	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5
byte 3	ID4	ID3	ID2	ID1	ID0	RTR	0	0
byte 4	DATA1							
:	:							
byte 11	DATA8							
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

ID - message identifier
RTR - data request
0 - common data message
1 - one of the nodes of the CAN bus requires data sending with the identifier carried within the message
DATA1 - DATA8 - message data
0 to 8 bytes according to the DLC value in the *Frm* variable

Sent messages

Items *Dats1* to *Dats32* containing received messages each have the following structure TCSJ_DATS:

Cont - message control (8-times bool type)

	0	0	0	0	0	0	0	SEND
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

SEND - data sending
0 - message will not be send
1 - send the message (bit is automatically reset after the message enqueuing)

Frm - message format (8-times bool type)

	FF	RTR	0	0	DLC3	DLC2	DLC1	DLC0
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

DLC3 - DLC0 - data length
Length of data carried in the message in the range of 0 to 8.
RTR - data request
0 - common data message
1 - data request from another node with the identifier carried in the message
FF - message frame format
0 - standard frame (identifier of the width of 11 bits)
1 - extended frame (identifier of the width of 29 bits)

Data [x] - message identifier and data (usint item array type)
The structure depends on the frame format
Standard frame:

byte 0	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
byte 1	ID2	ID1	ID0	RTR	0	0	0	0
byte 2	DATA1							
:	:							
byte 9	DATA8							
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

Extended frame:

byte 0	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
byte 1	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
byte 2	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5
byte 3	ID4	ID3	ID2	ID1	ID0	RTR	0	0
byte 4	DATA1							
:	:							
byte 11	DATA8							
bit	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

ID - message identifier

RTR - data request

The value of this bit is not vital, however, the producer of the sequencer recommends its value to be agreed with the value of the RTR bit in the variable *Frm*.

DATA1 - DATA8 - message data

0 to 8 bytes according to the DLC value in the variable *Frm*

2.13.2.2. Data interchange

Reception of data sent by another node

Into the receiving stack, there are all received messages stored, that are transmitted by the receiving filter (settings, see chapter 2.13.1.), in the order as they were received. The receiving stack allows to receive up to 32 messages during the user program cycle. Received messages are during the cycle scan transferred to the scratchpad and published for the user program within items of the receiving zone *Datr1* to *Datr32*.

The number of received messages contains the *Mesr* variable. If its value is 5, it means that 5 messages were received and were saved within items *Datr1* to *Datr5* in the order as they were received.

If there is more than 32 messages received during the cycle, the messages that did not get in the stack, will be lost. The number of these lost messages contains the *Meso* variable. If this event occurs, we must shorten the period of PLC cycle or reduce the operational intensity within the CAN bus or change the settings of the receiving filter so that less messages is passed through it.

Data request from another node

All messages that carry data have the RTR bit set to log 0. If a message with the RTR bit set to log.1 comes then it carries no data or conversely, it requires sending of data relevant to the identifier which this appeal came with. The user program should react by sending the message with following data.

Example:

received message:	ID = $0401, RTR = 1
sent message:	ID = $0401, RTR = 0, DATA

Standard RTR bit value is always in the variable *Frm*.

Data sending to another node

Into the sending zone of the scratchpad into items *Dats1* to *Dats32*, we enter messages that we want to be send.

The number of valid messages to be send contains the *Mess* variable. If its value is 5, it means that 5 messages are ready to be send within items *Datr1* to *Datr5* that will be send in this order.

The sending of each message is also conditioned by setting of the SEND bit to log.1 within the *Cont* variable in the message structure. This bit is erased within the cycle scan in the moment of message enqueuing to the communication channel. If the sequencer is in the reset mode, then the message is not enqueued and the SEND bit stays set to log.1.

The number of non-sent messages within the sending queue contains the *Mest* variable. It is a value detected at the input to the cycle scan, so before adding of new messages that are to be send. Therefore, it is a number of previous messages that still wait in the queue to be send. If this value increase constantly, it means that the sequencer is not able to send those messages so fast. The internal sending queue is able to hold 255 messages. It is solved by means of a circle stack, it means that if the stack is overloaded then with the entry of the 256th message, it will be reset and all messages including the newly arrived one will be lost. The following message will be send as usual again.

It is possible to approach the sending zone by two methods thanks to the SEND bit. We can either use items *Dats1* to *Dats32* as a classical stacks, meaning that during each cycle we will enter messages that are to be send into them, set to all of them relevant SEND bits to log.1 and enter their number into the variable *Mess*.

Or we also can use items *Dats1* to *Dats32* statically, meaning that each item is dedicated for messages with a concrete identifier. In this case, we have messages permanently prepared in items *Dats1* to *Dats32,* and within each cycle we refresh their data and set relevant SEND bits to log.1 periodically (e. g. every 200 ms and each in a different cycle). Into *Mess* variable we enter the number of all such operated items, not the number of currently sending messages. This process is suitable for sending of periodical data objects especially. Both stated processes can be combined, e. g. the first four items *Dats1* to *Dats4* can be reserved for periodical sending and items *Dats5* to *Dats32* are used as the classic message stack. Then, into *Mess* variable we enter the number of all messages saved to the sending zone regardless of the relevant SEND bits statuses. Therefore, it is the value correspondent with the highest index of item *Dats* used (e. g. if we save the last message in to the item *Dats18*, we enter the value 18 to the *Mess* variable).

Data loading from another node

If we want to read data from another node (station) that these data do not send actively, we must send a request presented as a message with the relevant identifier where the RTR bit is set to log.1. This message does not contain any data, however, the length of DLC data is usually set to the length value of data expected. The reason is to prevent the error on the bus when two CAN sequencers start to send the data request with the same identifier simultaneously. The same conditions apply for sending of these messages as for data messages (see previous paragraphs). The leading value of the RTR bit is always at the *Frm* variable.

2.13.2.3. Sequencer´s modes and error states

Sequencer modes

The SJA1000 sequencer runs in two basic modes - the reset mode and the operation mode. In the reset mode the sequencer is disconnected from the CAN bus and allows the setting of the communication parameters. In the operation mode the sequencer is connected to the CAN bus and executes communication according to the set communication parameters which can not be changed in this mode.

The sequencer gets to the reset mode by three methods:

- during the restart of the user program
- via the command from the user program
- at the high error occurance on the CAN bus

The sequencer gets to the operation mode by three methods:

- after the restart of the user program
- via the command from the user program
- after the automatic re-initialisation evoked by the high error occurance on the CAN bus

The sequencer status is indicated by bits in the *Statl* variable within the receiving zone. The RESM bit has the value of log.1 in the reset mode. Transient conditions between both modes are indicated by START and STOP bits.

The reset mode evoked by the user program

During the user program restart, the CAN sequencer is set to the reset mode. The sequencer is disconnected from the bus in this mode and allows the setting of the communication parameters. After the sequencer initialisation, the operation mode of the sequencer is actuated.

The sequencer can be set to the reset mode anytime using the RESM bit in the *Contl* variable within the sending zone. Until this bit has the value of log.1, the sequencer stays in the reset mode. Once the value of the RESM bit in the *Contl* variable changes to log.0, the sequencer is initialised again to values set by the user program and leaves the reset mode.

While conversion to the operation mode, the sequencer will wait for the idle time on the bus (the line has the idle time level of 11 bits).

Errors on the bus

The sequencer contains counters of bus errors detected during reception and sending. Their values are accessible through *Ecr* and *Ect* variables. The bus error is notified by the BE bit in the *Err* variable and the detailed diagnostics of this error is shown in the *Ecc* variable.

If any of both counters reach the value of 96, there is the EW bit set to log.1 in the *Err* variable. After the value of 255 on the error counter is reached during the sending, there is the BS bit set to log.1 in the *Err* variable and the sequencer is disconnected from the bus and converts to the reset mode separately. In the following cycle scan the sequencer is re-initialised automatically and transferred back to the operation mode again.

In contrast to the leaving from the reset mode evoked by the controlled request (restart of the user program, restart of the user program sequencer), there is, during the transfer to the operation mode after the error occurance on the bus, the waiting period when the sequencer always waits for 128 idle time occurances on the bus (the line has the dwell time level of 11 bits). The number of idle time occurances on the bus is indicated by the *Ect* counter by subtracting from the value 127 to 0. The *Ecr* counter is reset automatically.

Two independent CAN buses

The MR-0160 submodule contains a pair of CAN sequencers that are absolutely independent on each other. All information described above apply for each sequencer separately. In the scratchpad, there are structures for each CAN bus separately.

PLC detects the used module type. If we use the MR-0161 submodule which contains one CAN sequencer only then any contingent settings of the second CAN bus in the user program is ignored.

3. EPSNET NETWORK

Two types of stations can work on the EPSNET network:

master station (master) - active participant, controls communication
slave station (slave) - passive participant, responses to the queries of the master station

Communication is initiated by the master station on the query - response basis. This principle allows the connection of higher number of participants to the master station on the semi-duplex interface RS-485.

The time relations of the EPSNET network allow also the usage of various types of modems (except the special version EPSNET-F network).

3.1. EPSNET NETWORK CONFIGURATION

The EPSNET network allows two basic configurations:

monomaster	- there is one master station and several slave stations in the network
multimaster	- there are several master stations and several slave stations in the network

3.1.1. EPSNET monomaster configuration

The network contains one master station and several slave stations. The number of slave stations is usually limited by the transmission media used, the maximum is 126.

The network operation is controlled by one master station only.

This most frequently used method of communication allows data exchange between the master station and any of the slave stations. If we need to exchange data between two slave stations, we must exchange data through the master station.

The master station is usually represented by a PC with the visualisation or control software equipped with a driver for communication with systems TECOMAT / TECOREG or systems TECOMAT / TECOREG with the serial channel in the **MAS** mode or in the **MPC** mode with token message transmission switched off (parameter MT = off). The slave stations are systems TECOMAT / TECOREG with the serial channel in the **PC** mode.

3.1.2. EPSNET multimaster configuration

The network contains several master stations and several slave stations. The number of master stations is not in total number limited. The number of all stations is usually limited by the transmission media used, the maximum is 127.

The network operation at the given moment is controlled by one master station and after request completion, it passes control to the next master station. Slave stations do not participate in network control.

All other stations including master stations behave as slave stations towards the station currently controlling network operation.

This way of communication allows data exchange between the master station and any of the slave stations and among the master stations.

If we need to exchange data between two slave stations, we must exchange data through the master station. However, if one of these stations is re-configured to the master station then data can be exchanged directly.

Master stations are systems TECOMAT / TECOREG with the serial channel in the **MPC** mode (token message transmission must be switched on - parameter MT = on). Slave stations are systems TECOMAT / TECOREG with the serial channel in the **PC** mode.

The master station can be also a PC with the visualisation or control software equipped with a driver for communication with systems TECOMAT / TECOREG within the **multimaster** mode.

Attention! If it is not expressly given in driver description that it supports the **multimaster** mode, it is a **monomaster** driver and it must not be used in this configuration!

3.1.3. EPSNET-F version

A special case of master configuration is the EPSNET-F version. In this case the network contains only slave stations. Their number is limited by the possibilities of individual stations (16 to 64, for details see the **PLC** mode description valid for the given system).

Master stations are systems TECOMAT / TECOREG with the serial channel in the **PLC** mode.

The EPSNET-F network represents a modification optimised for data exchange at the shortest possible time. Data sent by one station are received by all other stations at the same time, which means that a part of a memory is shared by all participating systems. Communication takes place at higher speeds (usually 115.2 - 230.4 kBd).

As the transmission medium for this version the RS-485 interface without repeaters exclusively is used.

The EPSNET-F version is usually used as an accessory to the EPSNET monomaster network when the EPSNET monomaster network on one serial channel serves for data exchange among the control PC and individual systems TECOMAT / TECOREG and the EPSNET-F network on the second channel is used for data exchange among individual systems. The advantage of this combination is a high data throughput and independence of both data flows, the disadvantage can be a limited number of transmitted data and the requirement on the second serial channel at all TECOMAT / TECOREG systems.

3.2. EPSNET PROTOCOL GENERAL STRUCTURE

Communication in direction master station → slave station (message, query)

a) Message without data array

SD1	DA	SA	FC	FCS	ED

b) Message with data array

SD2	LE	LER	SD2R	DA	SA	FC	DATA.....	FCS	ED

c) Message token - network control take-over (without response, control take-over is the response)

SD4	DA	SA

Communication in direction slave station → master station (acknowledgment, response)

a) Response without data array

- short positive acknowledgment

SAC

- another type of acknowledgment

SD1	**DA**	**SA**	**FC**	**FCS**	**ED**

b) Response with data array

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**DATA.....**	**FCS**	**ED**

SD1 - start delimiter 1
- fixed value $10

SD2 - start delimiter 2
- fixed value $68

SD4 - start delimiter 4
- fixed value $DC

LE - data length
- number of bytes of items DA+SA+FC+DATA, i.e. 3 ... 249

LER - length repeat
- the same value as LE

SD2R - repeated start delimiter 2
- fixed value $68

DA - destination address
- value for slave station 0 ... 126 (query)
- value for master station 0 ... 126 (response)

SA - source address
- value for master station 0 ... 126 (query)
- value for slave station 0 ... 126 (response)

FC - frame control byte
- concrete values are given within particular messages (chapter 3.7.)

DATA- message data body
- maximum 246 bytes

FCS - frame check sum
- byte sum of all bytes of item DA, SA, FC a DATA with ignoring higher orders created by transmission

for SD1 $\sum_{DA}^{FC} \mathrm{mod}\, 256$

for SD2 $\sum_{DA}^{FCS-1} \mathrm{mod}\, 256$

ED - end delimiter (end delimiter)
- fixed value $16

SAC - short acknowledgement
- fixed value $E5

3.3. COMMUNICATION CONDITIONS

Conditions necessary to fulfil by the master station to perform the transmission

From the time point of view, the master station must follow the following conditions during communication:

- A **shorter** timeout (idle condition on the line) than the triple of the time needed for sending of one byte must be between the individual bytes being sent from the master station to the slave station

Note: The **PC** mode allows exception to this rule because it allows to set the space between received characters up to 255 ms. Therefore, such cases can be solved when the transmission mediums (modems) cause the message ripping. The condition is that the first 8 message characters must be received gathered (so according to the above mentioned rule). Only after this can the message be interrupted.

- There must be an idle condition on the line **longer** than the triple of the time needed for sending of one byte (fig. 3.1) between the response received and the next sent message, by means of the idle condition on the line, the receivers of all stations are synchronised before new transmission.

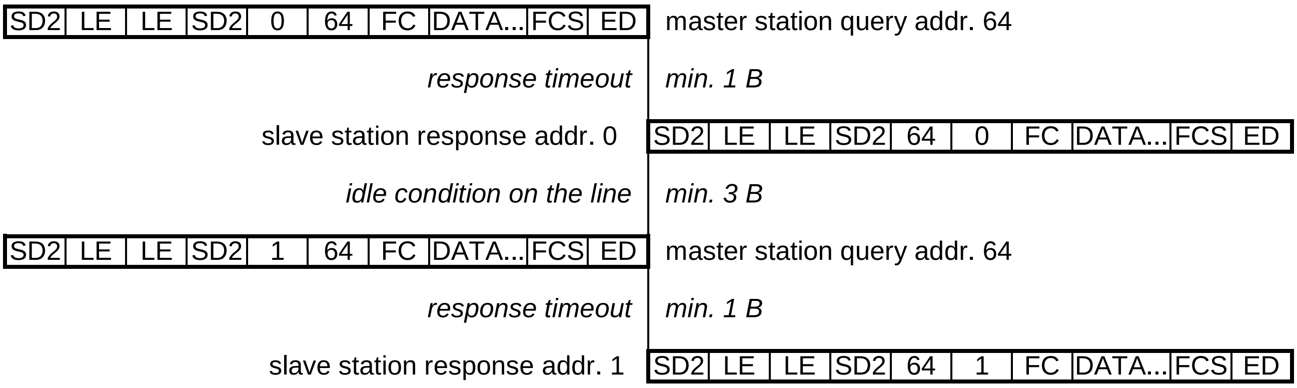

SD2 | LE | LE | SD2 | 0 | 64 | FC | DATA... | FCS | ED
master station query addr. 64
response timeout
min. 1 B
slave station response addr. 0
SD2 | LE | LE | SD2 | 64 | 0 | FC | DATA... | FCS | ED
idle condition on the line
min. 3 B
SD2 | LE | LE | SD2 | 1 | 64 | FC | DATA... | FCS | ED
master station query addr. 64
response timeout
min. 1 B
slave station response addr. 1
SD2 | LE | LE | SD2 | 64 | 1 | FC | DATA... | FCS | ED

Fig.3.1. Time relations on the line

Further principles of communication

Attention should be paid to following principles:

- TECOMAT responses to each message, if not, then it means a critical communication error (incorrectly compiled message, impassable line, the above mentioned conditions are not followed);
- The timeout between the last byte of the sent message and the first byte of the received response is minimally the same as the time needed for sending of one byte (see above), the maximum timeout is determined by the programmer of the master station with regards to the transmission rate, route (for example when using radio modems, the communication delay time increases) and cycle period (data read and entered are performed only at the I/O-scan to avoid data time hazard during the user program execution);
- The address of the master station must be different from addresses of the other stations.

Principles for multimaster mode

For the multimaster mode following principles apply additionally:

- If there is no operation on the network, the master station can start operation without token, this time has to be longer than the longest timeout (the time between the query and the response of the slave station) and has to depend on the address of the master station (possibility of collision of the access of two stations when they are switched on at the same time) according to the following formula

  time without operation = timeout + 500 + 10*address [ms]

- After the token message is received, the called master station waits for a time a little bit longer than the triple of the time needed for sending of one byte and then it starts operation in the network as the master station (fig.3.2)
- After the requirements completion, it passes the token message to the next master station.

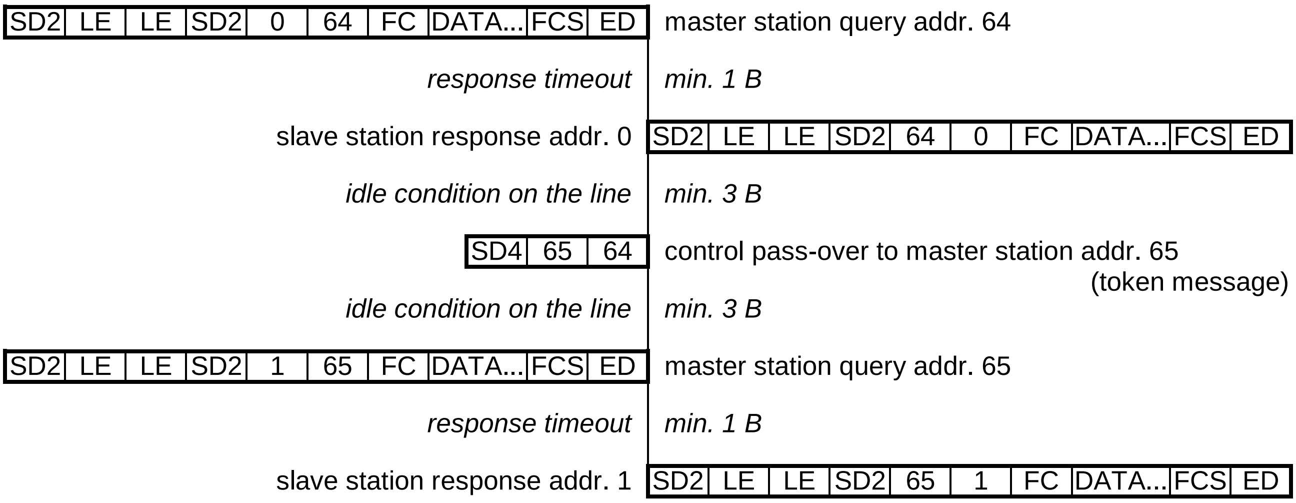

SD2 LE LE SD2 0 64 FC DATA... FCS ED
master station query addr. 64
response timeout
min. 1 B
slave station response addr. 0
SD2 LE LE SD2 64 0 FC DATA... FCS ED
idle condition on the line
min. 3 B
SD4 65 64
control pass-over to master station addr. 65
(token message)
idle condition on the line
min. 3 B
SD2 LE LE SD2 1 65 FC DATA... FCS ED
master station query addr. 65
response timeout
min. 1 B
slave station response addr. 1
SD2 LE LE SD2 65 1 FC DATA... FCS ED

Fig.3.2. Time relations on the line during control pass-over

Data format

All messages within the EPSNET network have a fixed format of 1 start bit, 8 data bits, even parity, 1 stop bit. Central units allow to switch off the parity in case of data transmission via modems which do not support parity (see chapter 3.4).

3.4. MESSAGE DATA PROTECTION

Standard data protection

Data in the message are protected by even parity, FCS check sum and correct sequence of the values of the protocol frame. Any error detected by these means causes that the message is declared invalid and is cancelled. Therefore, it is ensured that data passed on to the system are correct.

Parity switch off

In case we must transmit data via modems that do not support parity transmission, the TECOMAT central units allow to switch the parity off.

Additional protection with CRC polynomial

However, the security of data transmission is thus reduced and we must undertake an additional protection of transmitted data, preferably using a 16-bit CRC polynomial. We use system instruction CRC16 for the calculation of polynomial $x^{16} + x^{15} + x^{2} + 1$ for this purpose.

We add the calculated value CRC at the end of the data block transmitted. The character of the CRC value is that if we calculate the CRC value again through the same data including its CRC value being positioned behind the data block, the result will be 0.

Example for calculation and check of CRC

```
#def lengthr 12            ;length of received data including CRC
#def lengths 14            ;length of sent data including CRC
;
#reg usint messager[lengthr]
                      ;received data including CRC polynomial at data end
#reg usint messages[lengths]
                      ;sent data including CRC polynomial at the data end
;
P 0
     LD    __offset (messager)        ;start of received message
     LD    lengthr                    ;message length
     CRCM                             ;CRC check
                                      ;if A0 = 0, data is valid
:
     LD    __offset (messages)        ;start of sent message
     LD    lengths - 2                     ;message length
     CRCM                             ;CRC calculation
     LD    __offset (messages) + lengths - 2
     WRIW                             ;entry of CRC at message end
E 0
```

It results from the above mentioned that the length of the data transmitted increases by 2 bytes.

3.5. COMMUNICATION SERVICES

System and public communication services

The EPSNET network contains two groups of services for communication with TECOMAT and TECOREG systems. Public communication services allow data exchange, system communication services are used for programming and debugging of these systems.

Availability of system communication services

System communication services used by the Mosaic development environment are available on all communication channels in the **PC** mode. However, these system services can be used only on one channel at one time. The licence to the usage of system services obtains the communication channel that raise the request as the first one. The licence elapses 5 seconds after communication termination.

Availability of public communication services

Public communication services are available on all communication channels in the **PC** or **MPC** mode. An overview of public communication services is given in the table 3.1.

Table 3.1 Overview of public communication services of the EPSNET protocol

Service code		Title	Function
$08	8	SETTID	time setting
$0A	10	GETSW	status word reading
$0B	11	READN	data memory reading
$0C	12	WRITEN	data memory entry
$0D	13	WANDRN	data memory entry and reading
$0E	14	GETERR	error stack reading
$0F	15	READB	bits from data memory reading
$10	16	WRITEB	entry to bits of data memory
$90	144	READBD	destructive bits reading from data memory
$91	145	READND	destructive data memory reading
$93	147	WANDRND	data memory entry and destructive reading

3.6. RESPONSES AND ERROR MESSAGES

Positive response

As it has been already mentioned, a response is received to each query (providing the message transmission is executed without error). The response can be either positive or negative. The positive response means the service has been carried out correctly and data transmitted are valid. Data differ according to the concrete service used.

Negative response

There can be three types of negative responses. The central unit either does not know the service (FC = 2) or this communication service is not activated at this moment (FC = 3), or the data required is not available yet, but it can be expected that data will be available shortly (FC = 9), or the communication service has invalid parameters that cannot be interpreted correctly (FC = $0C).

SD1	**DA**	**SA**	**FC**	**FCS**	**ED**

FC = 2 - unknown communication service
FC = 3 - communication service is not activated at this moment
FC = 9 - data not yet accessible

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**ER1 ER2**	**FCS**	**ED**

LE = 5
LER = 5
FC = $0C - communication service invalid parameters
ER1, ER2 - the code of the last error in the local error stack (see the manual of the relevant system)

No response

If no response is received (eliminating hardware problems such as line interruption, power supply failure, etc. or incorrectly set transmission rate), it means the frame of the message sent is not correct (incorrect lengths LE, LER, incorrect value SD, SDR, ED, invalid address DA, invalid check sum FCS, etc.) or the principles given in chapter 13.3. were not followed (too long timeout between message bytes or too short timeout between the previous response and the next message, possibly invalid data format).

Error messages

Operation error messages of the serial channel are displayed in the variable *Err* in the diagnostics zone or in the receiving zone of the relevant station (according to communication mode). The message received, where an error occurred, is ignored and added to the number of invalid communication (variable *falseMes*). The reason for these errors can be a high level of interference, incorrectly connected communication line, incorrect parameter setting or a failure on the side of the master system.

$10 Invalid start delimiter SD

The value of the start delimiter of the message does not correspond to the defined value.

$11 Parity error

A message byte had invalid parity.

$12 Error of repeated data length LER

The value of repeated data length LER differs from the previous value LE.

$13 Repeated start delimiter error

The value of the repeated start delimiter SDR is not $68.

$14 SA source invalid address

The response comes from another station than the one queried.

$15 Transmission direction flag error in the FC control byte

The flag of data transmission at FC does not correspond to reality.

$17 Invalid value of FC control byte

The value of the FC control byte does not correspond to any function.

$18 Check sum error FCS

The received value of the check sum does not correspond to the value calculated.

$19 Invalid end delimiter ED

The value of the end delimiter is not $16.

$20 Invalid service

The service required is not implemented.

$50 The slave station does not response.

No response received from the slave station after elapsing of the time specified (0,5 s + transmission delay).

$54 The slave station responded incorrectly

The invalid answer came from the slave station (protocol error).

3.7. COMMUNICATION SERVICES DESCRIPTION

The following text describes formats of public communication services. The first group consists of services needed for communication initiation. They are used for connection establishment. The second group includes operation services allowing to detect the status of the system queried. The third group contains data services performing the data transmission itself.

3.7.1. Communication initialisation

CONNECT - connection establishment

Function:

Start of communication connected to the initialisation of the communication structures.

Syntax:

Message

SD1	DA	SA	FC	FCS	ED

FC = $49 or $69

Positive response

SD1	DA	SA	FC	FCS	ED

FC = 0

Example:

Communication establishment with the station with the set address 0. We assign the address for the master station, for example 126.

message - $10 $00 $7E $69 $E7 $16
response - $10 $7E $00 $00 $7E $16

IDENT - system identification

Function:

Recognition of information about the type of the system connected.

Syntax:

Message

SD1	DA	SA	FC	FCS	ED

FC = $4E or $6E

positive response

SD2	LE	LER	SD2R	DA	SA	FC	DATA.....	FCS	ED

FC = 0

DATA	- byte 0 - LIS - identification string length (usually 7) byte 1 - LIP - protocol implementation type length (always 1) byte 2 - LST - length of string determining the structure version (always 3) byte 3 - LSW - length of string determining the sw version (always 3) from byte 4 - central unit identification string (e.g. CP7002C) byte 4+LIS - character of protocol implementation (B) from byte 4+LIS+LIP - structure version string (e.g. 2.5) from byte 4+LIS+LIP+LST - software version string (e.g. 1.2)

Example:

System identification with the set value of 0 reading. The master station has the address of 126.

message - $10 $00 $7E $6E $EC $16
response - $68 $15 $15 $68 $7E $00 $00 | $08 $01 $03 $03 | $43 $50 $37 $30 $30 $32 $43 $20 | $42 | $32 $2E $35 | $31 $2E $32 | $B4 $16
Data array in ASCII CP7002C B2.51.2

3.7.2. Operational services

SETTID - time setup

Function:

Time data entry to the RTC time circuit. From this moment on, the system works with the new time settings.

Note: Using the instruction RDT, the synchronised time in milliseconds can be read. After the reception of the SETTID services, time is set and milliseconds are set to zero (this is not valid for the time in registers S5 to S12 where milliseconds are not set to zero - the time is continuous due to the continuity of time-dependent system functions).

Syntax:

Message

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**08**	**TIME**	**FCS**	**ED**

LE = $0B
LER = $0B
FC = $43 or $63
TIME - time data of the length of 7 bytes
1st byte - year (the last binary number 0 - 99, $00 - $63)
2nd byte - month (1 - 12, $01 - $0C)
3rd byte - day (1 - 31, $01 - $1F)
4th byte - hour (0 - 23, $00 - $17)
5th byte - minute (0 - 59, $00 - $3B)
6th byte - second (0 - 59, $00 - $3B)
7th byte - day in the week (1 - Monday, 2 - Tuesday, 3 - Wednesday, 4 - Thursday, 5 - Friday, 6 - Saturday, 7 - Sunday)

Positive response

SAC

Example:

Data and time entry 20.1.1996 6:55:00 Friday to the slave station with address 3.

message - $68 $0B $0B $68 $03 $7E $63 **|** $08 $60 $01 $14 $06 $37 $00 $05 **|** $A3 $16
response - $E5

SETCW - control word setup

Function:

Entry into the control word.

Attention!	This is a system service, thus, it is accessible only on communication channels supporting system services.

Syntax:

Message

SD2	LE	LER	SD2R	DA	SA	FC	09	CW_L	CW_H	FCS	ED

LE = 6
LER = 6
FC = $43 or $63
CW_L - control word lower byte

0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0

CW_H - control word upper byte

MOD	BLO	CLO	TPR	RES	0	0	CLE
7	6	5	4	3	2	1	0

CLE - request on error clearing (active at 1)
RES - system restart request (active at 1)
TPR - type of restart
0 - warm restart
1 - cold restart
CLO - request on outputs clearing (active at 1)
BLO - request on outputs blocking (active at 1)
MOD - required system mode
0 - HALT mode
1 - RUN mode
0 - obligatory entry of value 0

Positive response

SAC

Example:

Blocking of outputs of the slave station with address 3.

message - $68 $06 $06 $68 $03 $7E $63 **|** $09 $00 $40 **|** $2D $16
response - $E5

GETSW - status word reading

Function:

Status word reading.

Syntax:

Message

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**$0A**	**FCS**	**ED**

LE = 4
LER = 4
FC = $4C or $6C

positive response

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**SW_L**	**SW_H**	**FCS**	**ED**

LE = 5
LER = 5
FC = 8
SW_L - status word lower byte

X	X	X	X	X	X	X	X
7	6	5	4	3	2	1	0

SW_H - status word upper byte

MOD	BLO	X	X	ERS	X	X	ERH
7	6	5	4	3	2	1	0

ERH - critical error indication (active at 1)
ERS - other errors indication (active at 1)
BLO - outputs blocking indication (active at 1)
MOD - system mode indication
0 - HALT mode
1 - RUN mode
X - reserved

Example:

Status word of the slave station with address 3 reading.

message - $68 $04 $04 $68 $03 $7E $6C **|** $0A **|** $F7 $16
response - $68 $05 $05 $68 $7E $03 $08 **|** $00 $80 **|** $09 $16

The system is in the RUN mode with unblocked outputs without errors.

GETERR - error stack reading

Function:

Reading of the main error stack of the central unit.

Syntax:

Message

SD2	LE	LER	SD2R	DA	SA	FC	$0E	FCS	ED

LE = 4
LER = 4
FC = $4C nor $6C

positive response

SD2	LE	LER	SD2R	DA	SA	FC	DATAE	FCS	ED

LE = $23
LER = $23
FC = 8
DATAE - the error stack containing in total 8 error messages in the order from the oldest to the latest, each of the length of 4 bytes (see the manual of the relevant system), the total array length is 32 bytes

Example:

The main error stack of the slave station with address 3 reading.

message - $68 $04 $04 $68 $03 $7E $6C **|** $0E **|** $FB $16
response - $68 $23 $23 $68 $7E $03 $08 **|** $00 $00 $00 $00 **|** $00 $00 $00 $00 **|** $00 $00 $00 $00 **|** $00 $00 $00 $00 **|** $00 $00 $00 $00 **|** $00 $00 $00 $00 **|** $08 $00 $00 $00 **|** $80 $30 $11 $24 **|** $76 $16

The system reports the cycle time exceeding error (80 30 pcpc), preceded by a warning (08 00 00 00).

MASKCW - setting of individual bits of the control word

Function:

Setting of individual bits of the control word.

Attention! This is a system service, thus it is accessible only on communication channels supporting system services.

Syntax:

Message

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**$11**	**$C0_L$**	**$C0_H$**	**$C1_L$**	**$C1_H$**	**FCS**	**ED**

LE = 8
LER = 8
FC = $43 or $63
$C0_L$ - lower byte of the zero-mask of the control word
$C0_H$ - upper byte of the zero-mask of the control word
all bits of the control word with 0 on their position will be set to zero, the other bits will be retained (AND function)
$C1_L$ - lower byte of the one-mask of the control word
$C1_H$ - upper byte of the one-mask of the control word
all bits of the control word with 1 on their positions will be set, others will be retained (OR function)

positive response

SAC

The structure of the control word is as follows:

CW_L - control word lower byte

0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0

CW_H - control word upper byte

MOD	BLO	CLO	TPR	RES	0	0	CLE
7	6	5	4	3	2	1	0

CLE - request on error clearing (active at 1)
RES - system restart request (active at 1)
TPR - type of restart
0 - warm restart
1 - cold restart
CLO - request on outputs clearing (active at 1)
BLO - request on outputs blocking (active at 1)
MOD - required system mode
0 - HALT mode
1 - RUN mode
0 - system reserve, do not set!

Example:

Blocking of the outputs of the slave station with address 3 with leaving the current mode without the necessity of its identification.

message - $68 $08 $08 $68 $03 $7E $63 | $11 | $FF $FF | $00 $40 | $33 $16
response - $E5

3.7.3. Data services

READN - data memory reading

Function:

Reading of the content of the scratchpad memory and of the content of the DataBox secondary memory.

Syntax:

Message

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**$0B**	**TR1**	**IR1$_L$**	**IR1$_H$**	**LR1**	**...**

...	**TRn**	**IRn$_L$**	**IRn$_H$**	**LRn**	**FCS**	**ED**

LE = 4 + 4n (n is the number of read data blocks)
LER = 4 + 4n
FC = $4C or $6C
TRn - scratchpad area, from which the reading is performed
0 - registers X
1 - registers Y
2 - registers S
3 - registers R
- DataBox secondary memory area from which the reading is performed
$80 - addresses $00000 - $0FFFF
$81 - addresses $10000 - $1FFFF
$82 - addresses $20000 - $2FFFF
etc. up to the maximum range of this memory (given by DataBox type)
IRn$_L$ - lower byte of index of the first read register
IRn$_H$ - upper byte of index of the first read register
LRn - number of read registers

Positive response

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**DATAR1**	**...**	**DATARn**	**FCS**	**ED**

LE = LR1 + ... + LRn + 3
LER = LR1 + ... + LRn + 3
FC = 8
DATARn - block of read registers

Example:

Reading of the content of registers R30 to R35 and X0 and X1 of the slave station with address 4.

message - $68 $0C $0C $68 $04 $7E $6C | $0B | $03 $1E $00 $06 | $00 $00 $00 $02 | $22 $16
response - $68 $0B $0B $68 $7E $04 $08 | $01 $02 $03 $04 $05 $06 | $01 $02 | $A2 $16

WRITEN - data memory entry

Function:

Writing to the registers of the scratchpad memory and to the DataBox secondary memory.

Syntax:

Message

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**$0C**	**TW1**	**IW1**$_L$	**IW1**$_H$	**LW1**	**DATAW1**	**...**

...	**TWn**	**IWn**$_L$	**IWn**$_H$	**LWn**	**DATAWn**	**FCS**	**ED**

LE = 4 + 4n + LW1 + ... + LWn (n is the number of written data blocks)
LER = 4 + 4n + LW1 + ... + LWn
FC = $43 or $63
TWn - scratchpad area where the entry is performed
0 - registers X
1 - registers Y
2 - registers S
3 - registers R
- DataBox secondary memory area where the entry is performed
$80 - addresses $00000 - $0FFFF
$81 - addresses $10000 - $1FFFF
$82 - addresses $20000 - $2FFFF
etc. up to the maximum range of the memory (given by DataBox type)
IWn$_L$ - lower byte of index of first written register
IWn$_H$ - upper byte of index of first written register
LWn - number of written registers
DATAWn - block of written registers

Positive response

SAC

Example:

Entry to registers R30 to R35 and Y0 and Y1 of the slave station with address 4.

message - $68 $14 $14 $68 $04 $7E $63 **|** $0C **|** $03 $1E $00 $06 $01 $02 $03 $04 $05 $06 **|** $01 $00 $00 $02 $01 $02 **|** $33 $16
response - $E5

WANDRN - entry and reading from data memory

Function:

Writing to registers and reading of the content of the registers of the scratchpad memory or the DataBox secondary memory.

Syntax:

Message

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**$0D**	**TR**	**IR_L**	**IR_H**	**LR**	**TW**	**IW_L**	**IW_H**	**LW**
DATAW			**FCS**	**ED**											

LE = $0C + LW
LER = $0C + LW
FC = $4C or $6C
TR - scratchpad area from which the reading is performed
0 - registers X
1 - registers Y
2 - registers S
3 - registers R
- DataBox secondary memory area from which the reading is performed
$80 - addresses $00000 - $0FFFF
$81 - addresses $10000 - $1FFFF
$82 - addresses $20000 - $2FFFF
etc. up to the maximum range of the memory (given by DataBox type)
IR_L - lower byte of index of the first read register
IR_H - upper byte of index of the first read register
LR - number of read registers
TW - scratchpad area where the entry is performed
0 - registers X
1 - registers Y
2 - registers S
3 - registers R
- DataBox secondary memory area where the entry is performed
$80 - addresses $00000 - $0FFFF
$81 - addresses $10000 - $1FFFF
$82 - addresses $20000 - $2FFFF
etc. up to the maximum range of the memory (given by DataBox type)
IW_L - lower byte of index of first written register
IW_H - upper byte of index of first written register
LW - number of written registers
DATAW - block of written registers

Positive response

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**DATAR**	**FCS**	**ED**

LE = LR + 3
LER = LR + 3
FC = 8
DATAR - block of read registers

Example:

Entry to registers R30 to R35 and reading of registers X0 and X1 of the slave station with address 4.

message	- $68 $12 $12 $68 $04 $7E $6C | $0D | $00 $00 $00 $02 | $03 $1E $00 $06 $01 $02 $03 $04 $05 $06 | $39 $16
response	- $68 $05 $05 $68 $7E $04 $08 | $01 $02 | $8D $16

READB - bits from data memory reading

Function:

Reading of bits content of the scratchpad memory.

Syntax:

Message

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**$0F**	**TR1**	**IR1$_L$**	**IR1$_H$**	**BR1**	**...**

...	**TRn**	**IRn$_L$**	**IRn$_H$**	**BRn**	**FCS**	**ED**

LE = 4 + 4n (n is the number of read bits)
LER = 4 + 4n
FC = $4C or $6C
TRn - scratchpad area from which the reading is performed
0 - registers X
1 - registers Y
2 - registers S
3 - registers R
IRn$_L$ - lower byte of index of read register
IRn$_H$ - upper byte of index of read register
BRn - index of read bit (0 to 7)

Positive response

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**BITR1**	**...** **...**	**BITRn**	**FCS**	**ED**

LE = 3 + n
LER = 3 + n
FC = 8
BITRn - value of read bit expanded to byte (0 → 00, 1 → $FF)

Example:

Reading of the content of bits R34.2 and R34.5 of the slave station with address 4.

message - $68 $0C $0C $68 $04 $7E $6C **|** $0F **|** $03 $22 $00 $02 **|** $03 $22 $00 $05 **|** $4E $16
response - $68 $05 $05 $68 $7E $04 $08 **|** $00 **|** $FF **|** $89 $16

The bit R34.2 contains the value of 0, bit R34.5 contains the value of 1.

WRITEB - entry to bits of data memory

Function:

Writing to bits in the scratchpad memory.

Syntax:

Message

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**$10**	**TW1**	**IW1$_L$**	**IW1$_H$**	**BW1**	**...**

...	**TWn**	**IWn$_L$**	**IWn$_H$**	**BWn**	**FCS**	**ED**

LE = 4 + 4n (n is the number of written bits)
LER = 4 + 4n
FC = $43 or $63
TWn - scratchpad area where the writing is performed
0 - registers X
1 - registers Y
2 - registers S
3 - registers R
IWn$_L$ - lower byte of index of written register
IWn$_H$ - upper byte of index of written register
BWn - index of written bit, the value written in the highest bit
(entry 0 - 00 to 07, entry 1 - $80 to $87)

Positive response

SAC

Example:

Setting of the bit R15.6 and clearing of the bit R17.1 of the slave station with address 4.

message - $68 $0C $0C $68 $04 $7E $63 **|** $10 **|** $03 $0F $00 $86 **|** $03 $11 $00 $01 **|** $A2 $16
response - $E5

READBD - destructive reading of bits from data memory

Function:

Reading of the content of the bits of the scratchpad memory and their subsequent clearing.

Syntax:

Message

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**$0F**	**TR1**	**$IR1_L$**	**$IR1_H$**	**BR1**	**...**

...	**TRn**	**IRn_L**	**IRn_H**	**BRn**	**FCS**	**ED**

LE = 4 + 4n (n is the number of read bits)
LER = 4 + 4n
FC = $4C or $6C
TRn - scratchpad area from which the reading is performed
0 - registers X
1 - registers Y
2 - registers S
3 - registers R
IRn_L - lower byte of index of read register
IRn_H - upper byte of index of read register
BRn - index of read bit (0 to 7)

positive response

SD2	**LE**	**LER**	**SD2R**	**DA**	**SA**	**FC**	**BITR1**	**... ...**	**BITRn**	**FCS**	**ED**

LE = 3 + n
LER = 3 + n
FC = 8
BITRn - value of read bit expanded to byte (0 → 00, 1 → $FF)

Example:

Reading of the content of bits R34.2 and R34.5 of the slave station with address 4.

message - $68 $0C $0C $68 $04 $7E $6C **|** $90 **|** $03 $22 $00 $02 **|** $03 $22 $00 $05 **|** $CF $16
response - $68 $05 $05 $68 $7E $04 $08 **|** $00 **|** $FF **|** $89 $16

Bit R34.2 contained the value of 0, bit R34.5 contained the value of 1. After reading both bits were set to zero.

READND - destructive reading from data memory

Function:

Reading of the content of registers of the scratchpad memory or the content of the DataBox secondary memory and their clearing.

Syntax:

Message

SD2	LE	LER	SD2R	DA	SA	FC	$0B	TR1	IR1$_L$	IR1$_H$	LR1	...	TRn	IRn$_L$	IRn$_H$	LRn	FCS	ED

LE = 4 + 4n (n is the number of read data blocks)
LER = 4 + 4n
FC = $4C or $6C
TRn - scratchpad area from which the reading is performed
0 - registers X
1 - registers Y
2 - registers S
3 - registers R
- DataBox secondary memory area from which the reading is performed
$80 - addresses $00000 - $0FFFF
$81 - addresses $10000 - $1FFFF
$82 - addresses $20000 - $2FFFF
etc. up to the maximum range of the memory (given by DataBox type)
IRn$_L$ - lower byte of index of the first read register
IRn$_H$ - upper byte of index of the first read register
LRn - number of read registers

Positive response

SD2	LE	LER	SD2R	DA	SA	FC	DATAR1	...	DATARn	FCS	ED

LE = LR1 + ... + LRn + 3
LER = LR1 + ... + LRn + 3
FC = 8
DATARn - block of read registers

Example:

Reading of the content of registers R30 to R35 and X0 and X1 of the slave station with address 4.

message - $68 $0C $0C $68 $04 $7E $6C | $91 | $03 $1E $00 $06 | $00 $00 $00 $02 | $A8 $16
response - $68 $0B $0B $68 $7E $04 $08 | $01 $02 $03 $04 $05 $06 | $01 $02 | $A2 $16

After the content reading the registers were reset.

WANDRND - data memory entry and destructive reading

Function:

Writing to registers and reading of the content of registers of the scratchpad memory or DataBox secondary memory with their subsequent clearing.

Syntax:

Message

SD2	LE	LER	SD2R	DA	SA	FC	$0D	TR	IR$_L$	IR$_H$	LR	TW	IW$_L$	IW$_H$	LW

DATAW	FCS	ED

LE = $C + LW
LER = $C + LW
FC = $4C or $6C
TR - scratchpad area from which the reading is performed
0 - registers X
1 - registers Y
2 - registers S
3 - registers R
- DataBox secondary memory area from which the reading is performed
$80 - addresses $00000 - $0FFFF
$81 - addresses $10000 - $1FFFF
$82 - addresses $20000 - $2FFFF
etc. up to the maximum range of the memory (given by DataBox type)
IR$_L$ - lower byte of index of the first read register
IR$_H$ - upper byte of index of the first read register
LR - number of read registers
TW - scratchpad area where the entry is performed
0 - registers X
1 - registers Y
2 - registers S
3 - registers R
- DataBox secondary memory area where the entry is performed
$80 - addresses $00000 - $0FFFF
$81 - addresses $10000 - $1FFFF
$82 - addresses $20000 - $2FFFF
etc. up to the maximum range of the memory (given by DataBox type)
IW$_L$ - lower byte of index of first written register
IW$_H$ - upper byte of index of first written register
LW - number of written registers
DATAW - block of written registers

Positive response

SD2	LE	LER	SD2R	DA	SA	FC	DATAR	FCS	ED

LE = LR + 3
LER = LR + 3
FC = 8
DATAR - block of read registers

Example:

Writing to registers R30 to R35 and reading of registers X0 and X1 of the slave station with address 4.

message - $68 $12 $12 $68 $04 $7E $6C **|** $93 **|** $00 $00 $00 $02 **|** $03 $1E $00 $06 $01 $02 $03 $04 $05 $06 **|** $8F $16
response - $68 $05 $05 $68 $7E $04 $08 **|** $01 $02 **|** $8D $16

After the content reading, the registers was reset.

4. COMMUNICATION CHANNELS

The summary description of communication channels is all interfaces that are used for serial data interchange among the PLC TECOMAT and other devices. These devices are serial channels (chapter 4.1.), Ethernet interface (chapter 4.2.) and USB interface (chapter 4.3.).

4.1. COMMUNICATION VIA SERIAL CHANNELS

Serial channels represent the basic communication interface. The physical interface (RS-232, RS-485, RS-422, etc.) is selected via changeable submodules. Some submodules are not the single transmission of the serial line but contain the sequencer which enables the connection to the relevant bus (e.g. CAN, PROFIBUS). However, if we talk consequentially about serial channels, we subsume also communication via these submodules.

4.2. COMMUNICATION VIA ETHERNET NETWORK

Ethernet is the well-established name of the local network using the TCP / IP protocol designed for fast data exchange among connected devices at speed of up to 10 Mbps to 100 Mbps. It is a network with the random access (CSMA). The network allows communication of systems on intra-company level or passing of data from technology directly up to the information level of plant control.

Ethernet in the PLC TC650 and TC700

Compact PLC TECOMAT TC650, central units CP-7002, CP-7003 and communication module SC-7102 PLC TECOMAT are standardly equipped with interface Ethernet 10baseT according to IEEE 802.3. This allows communication via standard LAN network with the TCP / IP transport protocol at speed of 10 Mbps. Communication includes both services required for PLC programming and services used by SCADA / HMI systems. The programming via Ethernet is supported by the Mosaic development environment from version 1.0.39.

The central unit CP-7004 TC700 is standardly equipped with interface Ethernet 10/100baseT according to IEEE 802.3. This allows communication via standard LAN network with the TCP / IP transport protocol at speed of 10 or 100 Mbps.

The connection to the Ethernet network is ensured directly by the RJ45 connector and the UTP cable according to the international standard IEEE 802.3. The IP address and the IP mask for communication within the network is set by means of the Mosaic development environment. The Ethernet channel on the central unit can be set using push-buttons, too.

The central unit CP-7005 is also equipped by the Ethernet interface that is used for synchronization of redundant systems and is not accessible to the user.

Ethernet in the PLC FOXTROT

Compact PLC TECOMAT FOXTROT are standardly equipped with interface Ethernet 10/100baseT according to IEEE 802.3. This allows communication via standard LAN network with the TCP / IP transport protocol at speed of 10 or 100 Mbps. Communication contain both services required for PLC programming and services used by SCADA / HMI systems.

The connection to the Ethernet network is ensured directly by the RJ45 connector and the UTP cable according to the international standard IEEE 802.3. The IP address and the IP mask for the communication in the network is set by means of the Mosaic development environment, at PLC equipped with the built-in display by buttons too.

Ethernet addressing

Each Ethernet channel has its IP address and IP mask. Their values depend on the setting of the off-side participant in communication. A general rule applies here that the IP addresses of both participants of communication must be identical within positions where the IP mask has a non-zero value. The IP mask should be identical for both participants, however, this is not conditioned.

Example:	PC		PLC	
	IP address:	192.168.1.1	IP address:	192.168.1.2
	IP mask:	255.255.255.0	IP mask:	255.255.255.0
	or:			
	PC		PLC	
	IP address:	192.168.12.1	IP address:	192.168.25.8
	IP mask:	255.255.0.0	IP mask:	255.255.0.0

Attention: The first IP address number **must not be 0**, the last IP address number **must not be 0 or 255!** The last IP mask number **must not be 255!** If bad IP address and IP mask values are entered, CPU sets these parameters to default values preset by the producer.

The IP address and the IP mask are valid for all modes used within one Ethernet channel. For the ETH1 channel, it is possible to set the IP address and the IP mask on the central using push-buttons. For both channels ETH1 and ETH2 (on the communication module), the IP address and the IP mask can be set through the Mosaic development environment in the Project manager in folder *Hw* **|** *HW Configuration* in the setting of the central unit and the communication modules (common table of communication channels setting - fig. 1.2).

Communication modes

Through the Ethernet interface it is possible to communicate with the EPSNET UDP protocol in **PC** and **PLC** modes. In the **UNI** mode it is possible to send and receive any data in the UDP packet. In the **MDB** mode it is possible to communicate with the MODBUS UDP and MODBUS TCP protocols.

The **PC** mode is permanently active and allows also PLC programming. It is possible to communicate with 4 to 6 participants at one time (according to module type – see table 4.2). However, system services at this moment can be used by one participant only. This restriction is applied to all system communication channels as the whole independently on the type or position of the channel.

The **MDB** mode is permanently active. It is possible to communicate with 2 participants at one time.

Moreover, it is possible to switch the **PLC** mode on for data exchange among the PLCs via Ethernet and the **UNI** mode for data exchange via UDP or TCP packet. The **UNI** mode allows the connection of 1 up to 8 participants (according to the module type – see table 4.2).

With the **PLC** and **UNI** mode on, function and communication possibilities in the **PC** and **MDB** mode are not affected.

Table 4.1 The list of modes on the Ethernet interface and their possibilities

Communication mode	Input port	Protocol	Activation
PC	61682	EPSNET UDP	permanently
PLC	61681	EPSNET UDP	initialisation by the user program
UNI	optional	general UDP, general TCP	initialisation by the user program
MDB	502	MODBUS UDP, MODBUS TCP	permanently

Table 4.2 The number of connections within individual communication modes

Communication mode Module	PC	PLC	UNI
TC650	4	1 broadcast	1 (from sw 1.6)
TC700 CP-7002, CP-7003	4	1 broadcast	1 (from sw 4.5)
TC700 CP-7004	6	1 broadcast	1 (to sw 3.9) 8 (from sw 4.0)
TC700 SC-7102	4 (to sw 3.1) 6 (to sw 3.2)	1 broadcast	8 (from sw 3.2)*
FOXTROT CP-1004, CP-1005	6	1 broadcast	1 (to sw 3.9) 8 (from sw 4.0)

Com. mode Module	PC	PLC	UNI	MDB
TC650	4	1 broadcast	1 (from sw 1.6)	0
TC700 CP-7002, CP-7003	4	1 broadcast	1 (from sw 4.5)	0
TC700 CP-7004	6	1 broadcast	1 (to sw 2.8) 8 (from sw 2.9)	2 UDP, 2 TCP (from sw 3.7)
TC700 SC-7102	4 (to sw 3.1) 6 (from sw 3.2)	1 broadcast	8 (from sw 3.2)*	0
FOXTROT CP-1004, CP-1005, CP-1014, CP-1015	6	1 broadcast	1 (to sw 2.8) 8 (from sw 2.9)	2 UDP, 2 TCP (from sw 3.7)

* In central units CP-7001, CP-7002, CP-7003, CP-7005 the version sw 4.7 is required

4.2.1. EPSNET UDP protocol structure

As it can be seen from the title, the EPSNET UDP protocol is based on the UDP protocol. The port number on the side of the TECOMAT PLC in the **PC** mode is always 61682. The **PLC** mode uses the port number 61681.

Data in the UDP packet sent to the PLC begin with the header of the length of 6 bytes, followed by the message itself that has the same structure as the standard EPSNET protocol (see chapter 3). Moreover, up to 5 EPSNET messages can be send beyond one header, which makes the communication more efficient (fig. 4.1). Then these multiple messages are processed by the PLC in such order as they are positioned.

The response from the PLC has the same format. The corresponding number of responses in the EPSNET protocol follows the header of the length of 6 bytes.

The logical address of the station (address of the EPSNET protocol) is 0. Other addresses 1 to 126 are used during the message transfer to another communication channel (see chapter 4.4.).

0 1	2	3	4 5	6 ...	
MESI	**PN**	**R**	**DPLEN**	**EPSNET1**	**EPSNET2**

EPSNET3	**EPSNET4**	**EPSNET5**

Fig.4.1 EPSNET UDP protocol structure

Heading structure

Each message of the EPSNET UDP protocol (in any direction) has a header at the start of the data zone formed by 6 bytes (fig. 4.1). They have the following meaning:

byte 0 and 1 **MESI** - session number
Serves for the identification of individual messages. The PLC returns in the response the same session number as it was carried by the query sent from the master system. Thus, it can be determined unequivocally which query the response received belongs to.

byte 2 **PN** - communication mode code
Determines which mode the following data should be processed in.
2 - **PC** mode (EPSNET slave)
3 - **PLC** mode (EPSNET multimaster)

byte 3 **R** - reserve

byte 4 and 5 **DPLEN** - length of data that follows
Specifies the length of following data (all EPSNET messages in this UDP packet without header). Byte 4 contains a higher length byte, byte 5 contains a lower length byte (Motorola convention).

Note: The total data length in the UDP packet must be compulsory even. If an uneven data length appears during communication, then it is necessary to add one zero-byte at the data end to increase the total data length in the UDP packet. The length modification mentioned above is necessary only for sending of the UDP packet, item DPLEN in the header is not modified (can be odd, too).

4.2.2. Communication between networks via gateway

If we need to connect the PC which is situated within other network than PLC TECOMAT, we interconnect both networks via so called gateway. The gateway has in each one from both networks the IP address assigned (fig. 4.2). The IP address of the gateway must be set in the PLC owing to the fact that messages are routed vie the gateway.

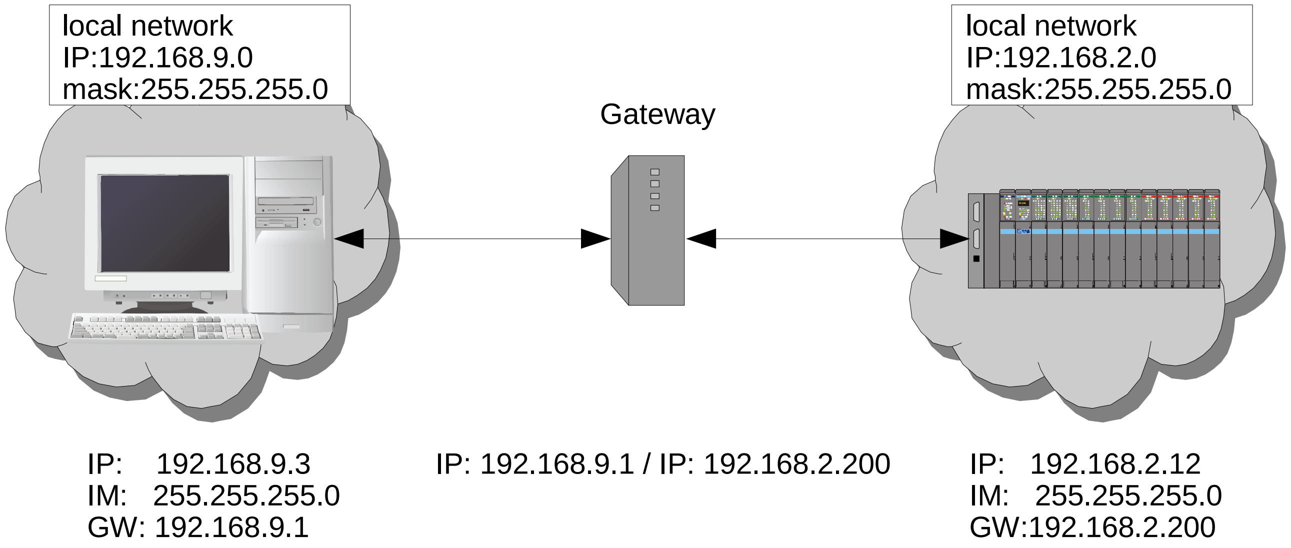
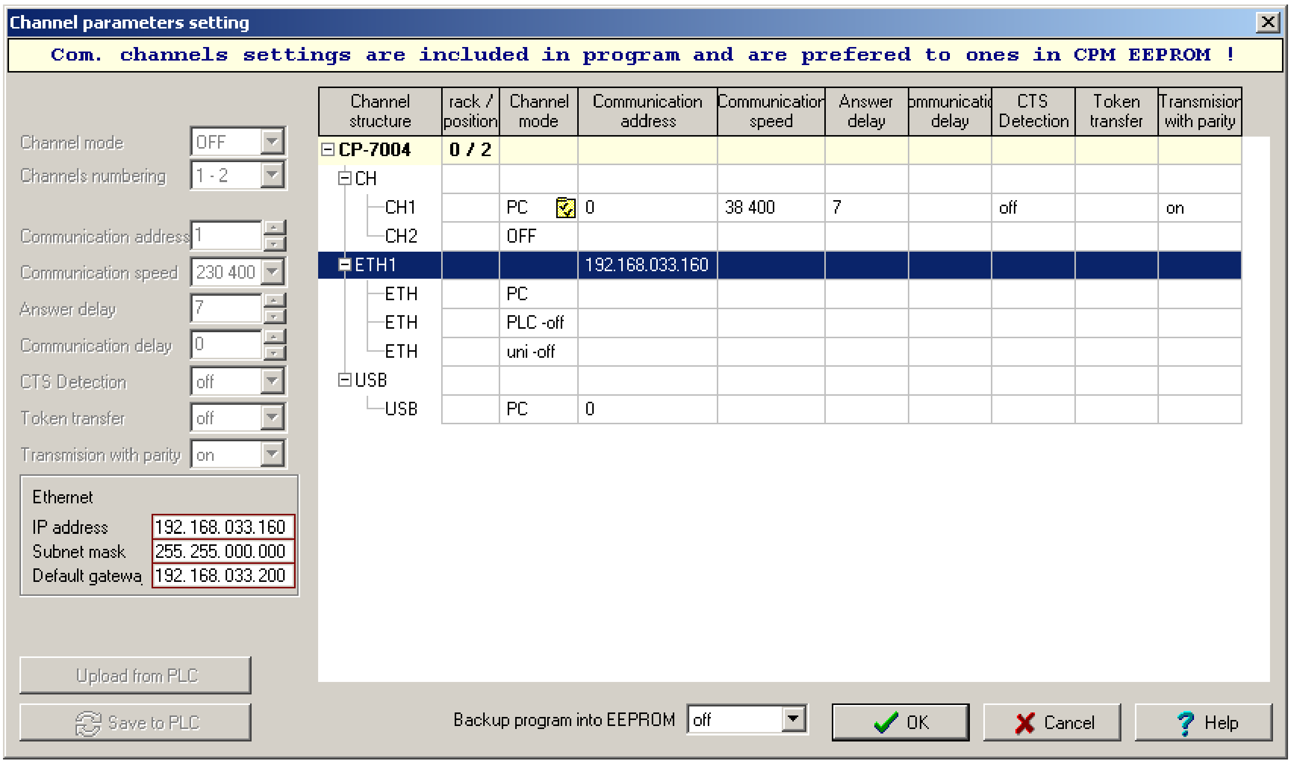


Fig.4.2 The example of interconnection of two networks via the gateway


Channel parameters setting

Com. channels settings are included in program and are prefered to ones in CPM EEPROM !

Channel mode OFF
Channels numbering 1 - 2
Communication address 1
Communication speed 230 400
Answer delay 7
Communication delay 0
CTS Detection off
Token transfer off
Transmision with parity on

Ethernet
IP address 192. 168. 033. 160
Subnet mask 255. 255. 000. 000
Default gatewa 192. 168. 033. 200

Channel structure	rack / position	Channel mode	Communication address	Communicatior speed	Answer delay	ommunicatio delay	CTS Detection	Token transfer	Transmisior with parity
⊟ CP-7004	0 / 2								
⊟ CH									
CH1		PC	0	38 400	7		off		on
CH2		OFF							
⊟ ETH1			192.168.033.160						
ETH		PC							
ETH		PLC -off							
ETH		uni -off							
⊟ USB									
USB		PC	0						

Backup program into EEPROM off

Fig.4.3 Setup of the IP address of the gateway in Mosaic

The setup of the gateway address in Mosaic is set in the Project manager, the Channel parameters setting of the central unit (fig. 4.3) into the parameter *Default gateway.* Set parameters must be entered to the PLC via the push-button *Save to PLC.* The IP address of the gateway can be entered to the central unit also by using push-buttons in the setup mode into parameters GW1 to GW4.

4.2.3. Communication between networks via Internet

PC and the remote PLC TECOMAT can be also interconnected via Internet using ADSL modems. To ensure that the interconnection of Internet and Ethernet is functional and secure, we must undertake several tasks described in paragraphs below.

Recommendation: Before changing the settings, read carefully through your firewall documentation. In case of company network, please contact your server administrator or other competent person. When the setup is set incorrectly, your local network can be accessible on the Internet!

Personal firewall setup

For successful communication within the Ethernet network, it is necessary to enable in the personal firewall PC UDP communication with the PLC. In most cases a personal firewall inquiry dialogue appears. It is necessary to check whether it is the UDP communication towards the PLC onto the 61682 port and enable it.

Source network firewall setup (i.e. where the Mosaic is ran)

It is necessary to enable in the firewall the outbound UDP communication from the local network to the Internet on the 61682 port. This applies in a case when in the firewall, there is not enabled any outbound communication.

Target network firewall setup (i.e. where the PLC is)

It is necessary to enable in the firewall the inbound UDP communication from the Internet to the local network on the 61682 port with IP address mapping (NAT) of the target PLC in the network. By the abbrev NAT (Network Address Translation) the so called IP address compilation is represented. It is the substitution of local IP addresses in place of public (internet) IP addresses in outbound packets from the local network and conversely, it is the substitution of public IP addresses in place of local IP addresses in inbound packets to the local network.

The settings of firewalls (NAT) differs from producer to producer. Some firewalls have communication rules, some (e.g. some ADSL modems) have the access set via the NAT setting.

It is required to set following items (the first two are not available within simpler firewalls, e.g. within some ADSL modems):

source IP address - concrete or optional IP address from Internet (i.e. from where will we connect from PC)

target IP address - public IP address of the target network (i.e. public IP network address where the PLC TECOMAT is connected to

public port - 61682 (on this port communication PC and PLC via Ethernet / Internet runs)

protocol - UDP (eventually TCP / UDP)

target address compilation - e.g. 192.168.2.161 (i.e. address of the target PLC within its own local network)

port compilation - 61682 (this item must be filled in some firewalls even though it does not differ from the public port)

At the end the newly created rule must be enabled.

The example is shown at fig.4.4.

On the PC with the IP address 192.168.9.3 the Mosaic environment runs. The interconnection device entering the local network under the IP address 192.168.9.1 has a public IP address 172.16.5.55. There must be enabled outbound communication via protocol UDP on the 61682 port in its firewall. In the Mosaic environment, must be the IP address 172.16.1.12 set, it is a public IP network address which the PLC is connected to.

PLC TECOMAT has the IP address 192.168.2.161. The interconnection device entering the local network under the IP address 192.168.2.200 has a public IP address 172.16.1.12. There must be enabled outbound communication via protocol UDP on the 61682 port in its firewall. This communication must be mapped (NAT) on the PLC with the IP address 192.168.2.161. On the PLC central unit, there must be set the item GW (IP address of the source gateway) to the value 192.168.2.200 which is the local IP address of the interconnection device.

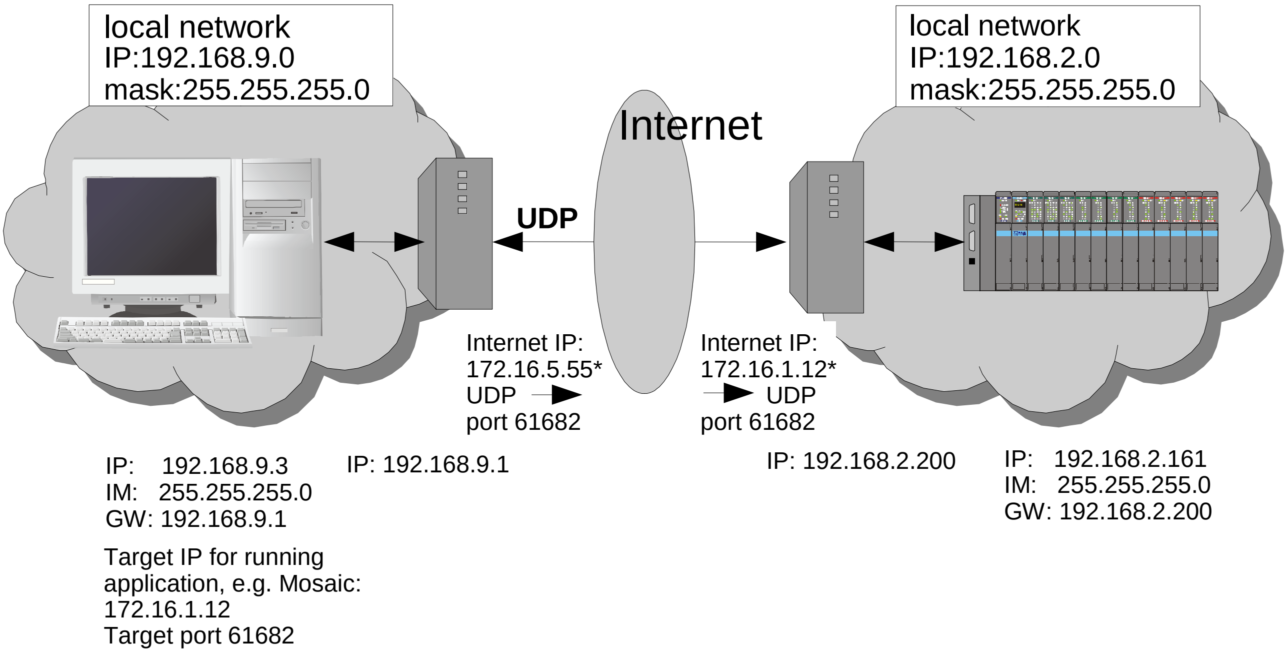


Fig.4.4 The example of interconnection of two networks via Internet
** these IP addresses are fictive as an illustration only, they can not be used within the Internet network*

4.3. COMMUNICATION VIA THE USB INTERFACE

The USB interface allows the connection of two co-operating devices, therefore, it can not be used within the network, and it is designed for the purposes of debugging, programming and servicing only. It must not be used for a stable connection onto the PC (e.g. for visualisation purposes of the controlled technology). The USB interface is not galvanically isolated!

The USB interface at the PLC TC700

The central unit is always mounted with one USB interface according to the USB 2.0 specification. The connector on the central unit corresponds to the USB, „B“ device specification.

For the PLC connection to the PC a standard USB A - B cable, max. 5 m long, twisted and shielded, can be used. The recommended KB-0208 cable is supplied under order number TXN 102 08.

Modes of communication

Via the USB interface it is possible to communicate in the **PC** mode. The **PC** mode is permanently active and allows also PLC programming, i.e. it supports system services (if they are not used by another communication channel at that moment).

The logical station address (EPSNET protocol address) is 0. Other addresses 1 to 126 are used during the message transfer to another communication channel (see chapter 4.4.).

PC driver

USB driver for communication in the Mosaic environment with the PLC is a part of the installation. After the Mosaic installation it is usually saved in the folder *C:\Program Files\Teco\Mosaic\UTIL\USB - Direct*. During the first connection of the PLC via USB, Windows announce the new hardware to be found and require the driver installation. After its installation communication via USB interface in the Mosaic environment is functional.

4.4. MESSAGE TRANSFER BETWEEN COMMUNICATION CHANNELS

There occur cases in the practise, when we have a realized PLC network, that are distributed within the technology but the access for tuning and programming is enabled only to one of them. It generally is the PLC that serves, among others, as a data concentrator for the controlling workplace. Central units CP-7001, CP-7002, CP-7003 TC700 (from sw version 4.7), CP-7004 TC700 and CP-10xx FOXTROT (all versions) and TC650 (from sw version 1.8) and communication modules SC-710x (from sw version 3.2) allow under certain conditions to transfer the message from one channel to another and make accessible also other PLC for tuning and programming which are not directly connected to tuning devices.

The basic condition is that the communication channel transferring the appeal to the slave PLC must be in the **PC** mode. The interface does not matter, it can be either the serial channel or USB interface or Ethernet. The communication channel accepting the appeal must be in the **MPC** mode.

Another condition is that both the transferring and the accepting communication channel must be on the same module. Therefore, it is not possible that one channel would be on the central unit and the second one on the communication module.

The third important condition is that addresses of all stations of all interconnected network must not collide. Here applies that the logical address of the central unit within the USB interface and Ethernet is always 0.

The interconnection is realized so, that we tick in the network parameters table in the **MPC** mode the box in the column marked by the icon on the line of the correspondent PLC, which we want to make accessible to the master station (chapter 2.4.2.). When the above terms are observed, we can, for example, make accessible the PLC from two networks simultaneously on CH1 and CH2 channels in the **MPC** mode for the master station via the USB interface or via Ethernet ETH1. The communication module SC-710x allows the interconnection of both its serial channels or of Ethernet ETH2 interface with one or both serial channels.

Only one interconnection to the same communication channel can be realized at one time. Therefore, it is not possible to pass on data from the USB and also from the Ethernet to CH2, however, it is possible to transfer data from the USB to CH1 and from the Ethernet to CH2.

The possibilities of communication channels interconnection are described in the table 4.3, 4.4, 4.5 a 4.6.

Table 4.3 Possibilities of communication channels interconnection within central units CP-700x TC700

Transferring channel in the PC mode	Accepting channel in the MPC mode	Note
CH1	CH2	
CH2	CH1	
USB	CH1, CH2	the channel address is always 0
ETH1	CH1, CH2	the channel address is always 0

Table 4.4 Possibilities of communication channels interconnection within communication modules SC-710x TC700

Transferring channel in the PC mode	Accepting channel in the MPC mode	Note
CH3 (CH5 / CH7 / CH9)	CH4 (CH6 / CH8 / CH10)	
CH4 (CH6 / CH8 / CH10)	CH3 (CH5 / CH7 / CH9)	
ETH2	CH3, CH4 (CH5, CH6 / CH7, CH8 / CH9, CH10)	the channel address is always 0

Table 4.5 Possibilities of communication channels interconnection within central units TC650

Transferring channel in the PC mode	Accepting channel in the MPC mode	Note
CH1	CH2, CH3	
CH2	CH1, CH3	
CH3	CH1, CH2	
ETH1	CH1, CH2, CH3	the channel address is always 0

Table 4.6 Possibilities of communication channels interconnection within central units CP-10xx FOXTROT

Transferring channel in the PC mode	Accepting channel in the MPC mode	Note
CH1	CH2	
CH2	CH1	
ETH1	CH1, CH2	the channel address is always 0

In practise, this function can be used according to the example in the fig. 4.5. In the Mosaic environment we set communication parameters valid for connection with the physically connected PLC. The selection of the PLC which we will communicate with, is then accomplished by the selection of the address 0, 1 or 2 only. As it results from mentioned conditions, the connection via Ethernet or USB can be realized even with systems of types NS950, TC400, TC500, TC600, TR050, TR200 and TR300 that are not fitted with these interfaces.

The only condition is that we activate for both stations 1 and 2 the forwarding selection in the network parameters table in the **MPC** mode.

MPC mode driver sends a message to the assigned station within the next nearest occasion (after the previous communication termination, or rather after acceptance of the network control from another master station). Communication timeout set in the Mosaic environment is therefore required to be selected with regard to the transmission delay on the network where the message is transferred to.

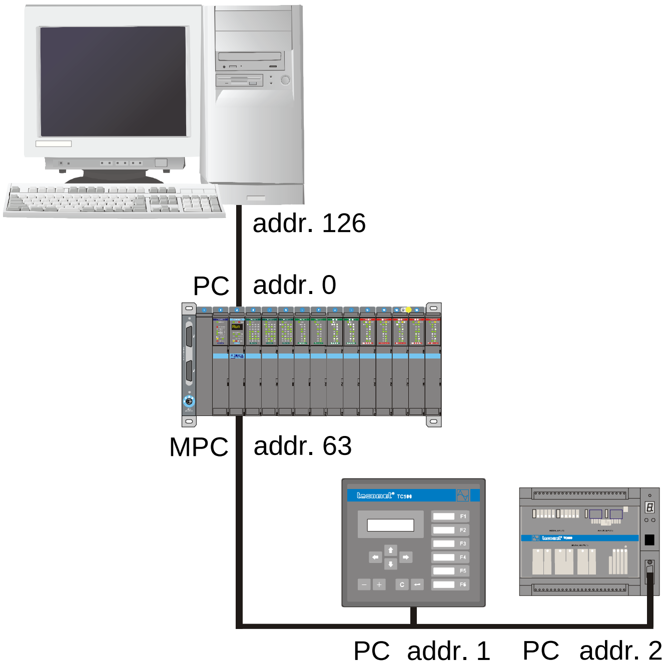


Fig.4.5 The example of the configuration using the message transfer between communication channels

5. COMMUNICATION EXAMPLES

5.1. COMMUNICATION WITH THE MASTER SYSTEM IN THE EPSNET NETWORK

This chapter describes some examples of PLC communication with the master system within the EPSNET network. The master system can be a PC or a TECOMAT PLC in the **MAS** or **MPC** mode.

In the following alternatives of connection, distances for the maximum transmission rate are specified. For non-switched data links, a connection with the conductor diameter of 0.65 mm with twisted pairs is taken into account. A detailed description of connection implementation can be found in the Manual for designing of TECOMAT PLCs, TXV 001 08.01.

5.1.1. Short distance connection

- Standard RS-232 interface
- No other additional means for connection required
- Transmission rate of up to 115.2 kBd
- Only for connection of one PLC (point-to-point connection) to a distance of 15 m

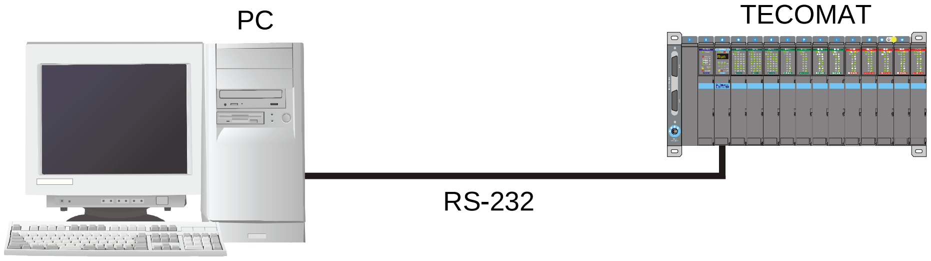


Fig.5.1 Connection to the master system on the RS-232 interface

5.1.2. Connection on a distance of hundreds of metres

- RS-485 serial interface
- For master systems equipped only with RS-232 interface, it is necessary to use a serial interface adaptor
- Transmission rate of up to 115.2 kBd
- Connection of up to 32 TECOMAT PLC to one master system on the line to a distance of 1.2 km

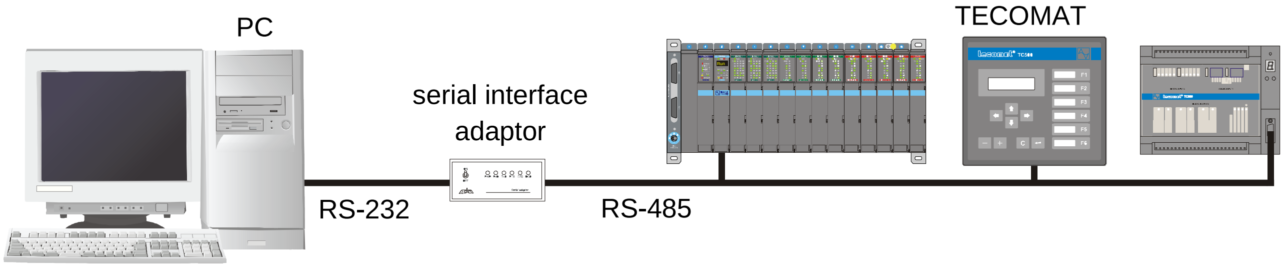


Fig.5.2 Connection to the master system on the RS-485 interface

5.1.3. Non-switched data link

- An alternative for distances in units of km at a favourable price
- A modem used for non-switched data link, such as SRM-5A, SRM-5D
- Max. transmission rate 19.2 kBd
- Max. length of non-switched data link 12 - 27 km

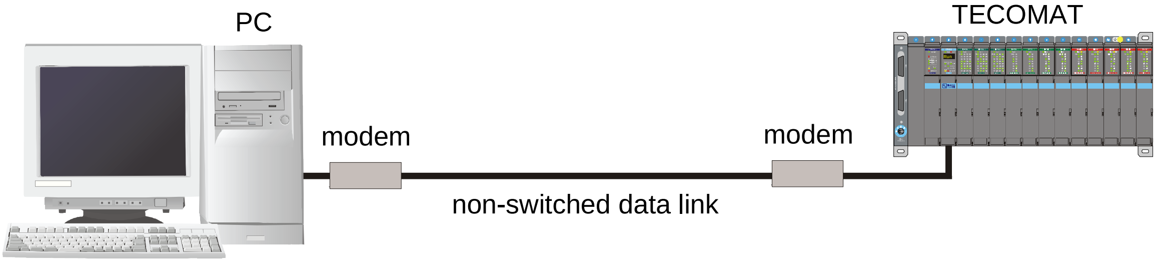


Fig.5.3 Connection to the master system via non-switched data link

5.1.4. Connection via telephone lines

- Designed for autonomously working PLC with occasional communication with the master system
- Possibility of phone number dialling of the master system from the PLC at exceptional or emergency situations
- The phone modem used allows setting of transmission parameters to 8 bits of data, even parity (not prerequisite, but recommended), 1 stop bit

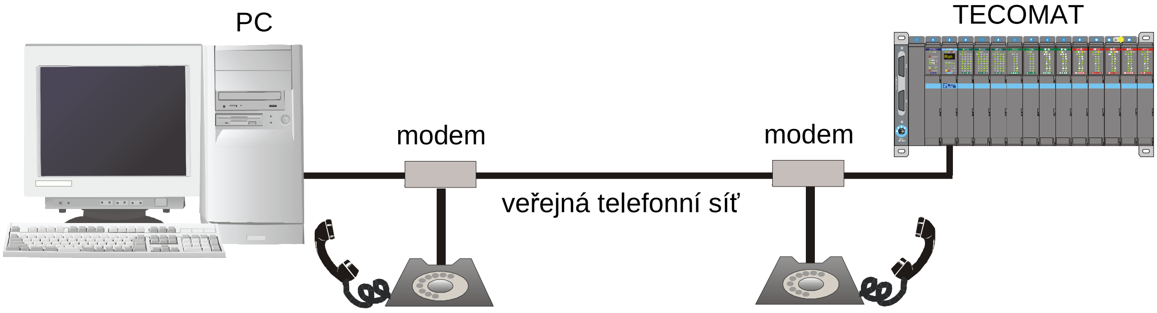


Fig.5.4 Connection to the master system via public telephone system

5.1.5. Connection via TV cable distribution

- Implementation of PLC networks communicating with the control room at urban concentration via TV cable distribution system
- Max. transmission rate 19.2 kBd
- Connection of up to 99 TECOMAT PLC to one master system
- Communication modems used: KM100 of company CATV Ltd.

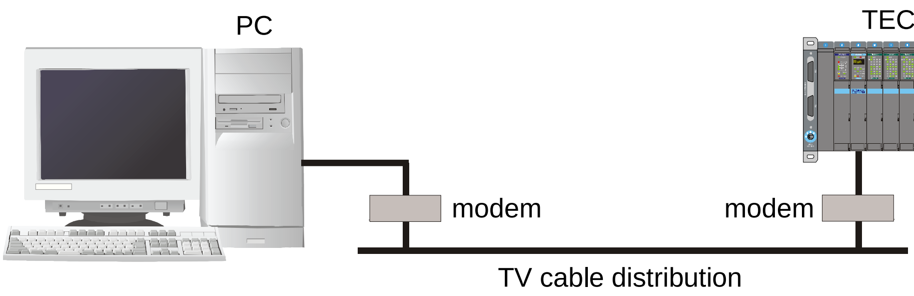


Fig.5.5 Connection to the master system via TV cable distribution

5.1.6. Wireless connection via radio modems

- Suitable anywhere where laying of cables for conventional connection is too expensive or technically impossible (e.g. the PLC used is part of a moving unit)
- Max. transmission rate 9.6 kBd
- Possibility of communication of more participants within the radio network
- Unit (complete) for radio connection used: Racom MR25

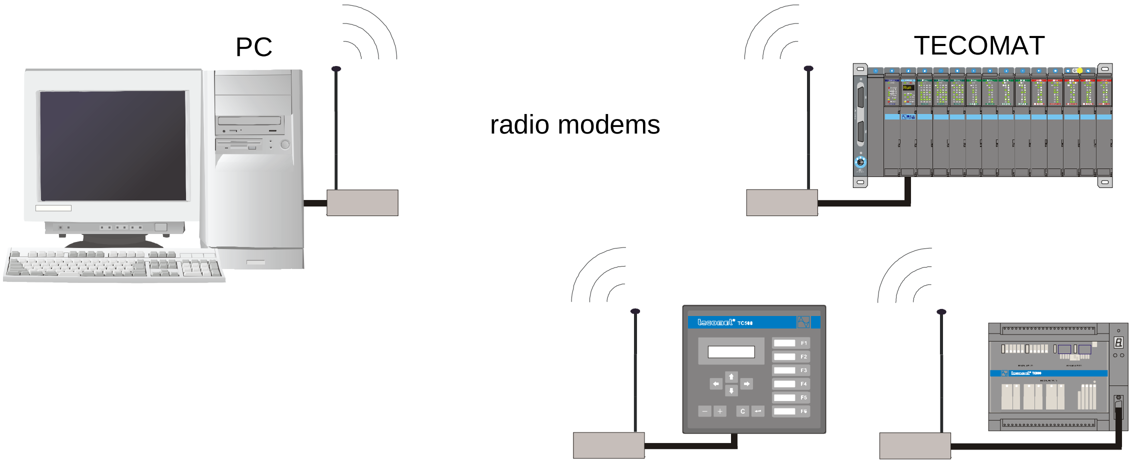


Fig.5.6 Connection to the master system via radio modems

5.1.7. Connection via power supply network

- Implementation of PLC networks communicating via 230 V power supply network
- Max. transmission rate 2.4 kBd
- Limitation of message size to 32 bytes including the frame, cannot be used for programming and debugging of PLCs by the Mosaic development environment
- Communication modems used: LONET, company ZPA Trutnov

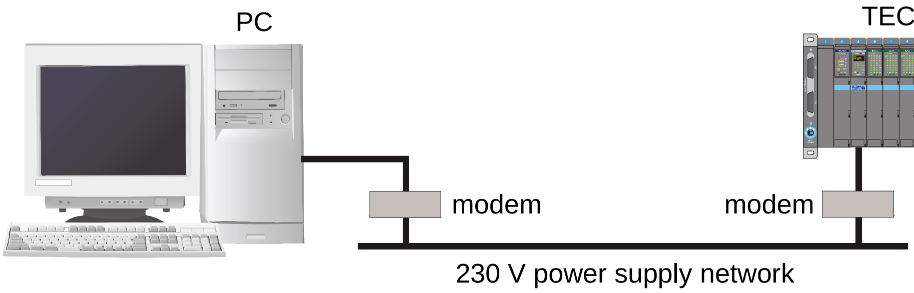


Fig.5.7 Connection to the master system via power supply network

5.1.8. Wireless connection via IR light

- Implementation of connection through spaces with a high level of interference, connection of two moving units
- Max. transmission rate 38.4 kBd
- Reach at direct visibility 0.2 to 180 m
- Implementation using converters IR / RS-232, RS-485, RS-422
- IR converters ISD230-4111, company SICK AG

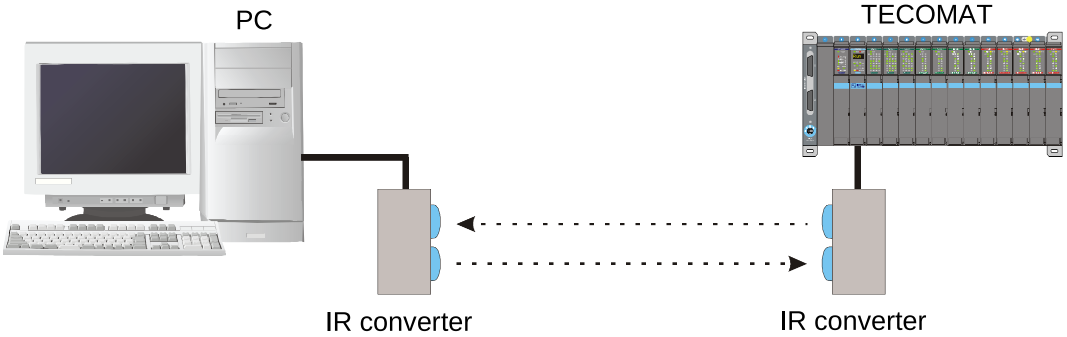


Fig.5.8 Connection to the master system via IR light

5.2. APPLICATION OF SMS WITHIN GSM NETWORKS

This way of communication is designed for autonomously working PLC with occasional communication with the GSM phone, the Internet SMS server or another control system. It allows SMS reception to the PLC and sending of SMS from the PLC.

For easier use of SMS services in the GSM networks, the *SMS* instruction can be used. This instruction allows the PLC to receive and send SMS while the instruction itself controls the communication with the GSM phone. By means of this instruction, we can send a query concerning the status of the technology being controlled from a common mobile phone or, in a case of a failure in the controlled technology or conversely, the PLC can send actively SMS to the mobile phone of the maintenance staff for intervention. The messages can be sent also among different GSM operators and thank to the interconnection of the GSM networks via e-mails, too.

The *SMS* instruction supports industrial GSM phones Siemens TC35 and their equivalents and requires a PLC serial channel in the **UNI** mode.

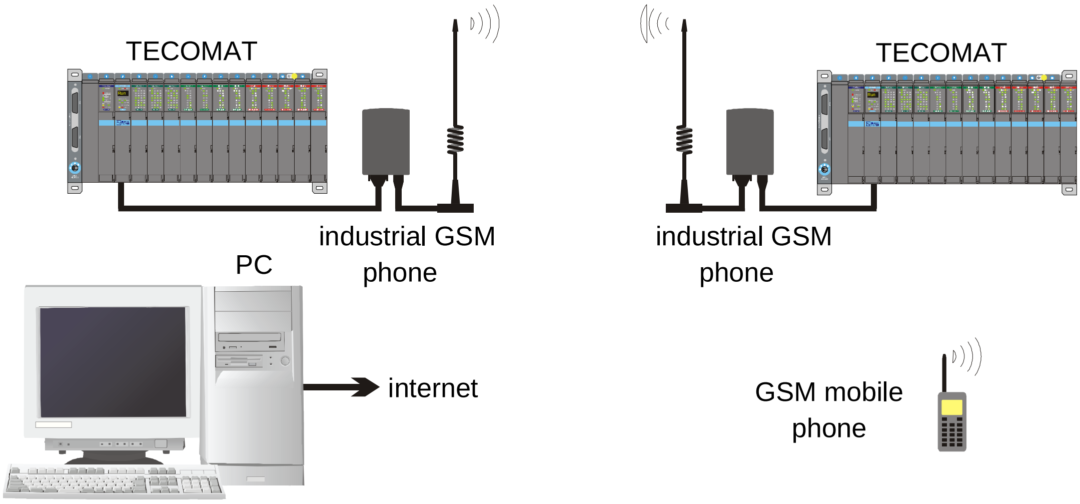


Fig.5.9 Connection of PLC to GSM network

For more information contact us :
Teco a. s. Havlíčkova 260, 280 58 Kolín 4, Czech Republic
tel. 321 737 611, fax 321 737 633, e-mail: teco@tecomat.cz, www.tecomat.com

We reserve the right to make modifications and/or changes of the documentation without prior notice.
The last issue is available at www.tecomat.com.