

SIMATIC TDC

System and communication configuring D7-SYS

System Manual

Preface

Just a few steps away from
the first project

1

System software

2

Configuring communication

3

Service & Support

A

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

CAUTION

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Preface

Purpose of this manual

This manual describes the principles in using the D7-SYS automation software and its functions, while setting the focus on the corresponding Technology and Drive Control components SIMATIC TDC or FM 458-1 DP.

TDC: Technology and Drive Control

Basic knowledge required

This manual addresses programmers and commissioning engineers. Comprehension of this manual requires general knowledge of automation engineering.

Scope of the manual

This manual is valid for SIMATIC D7-SYS as of version 8.2.

Position in the information landscape

This manual is part of the documentation package for the Technology and Drive Control components FM 458, SIMATIC TDC and SIMATIC D7-SYS.

Title	Contents
<p>System and communication configuration D7-SYS http://support.automation.siemens.com/WW/view/de/8776461/0/en</p>	<p>Just a few steps away from the first project This section provides an extremely simple introduction into the methodology of the structure and programming of the SIMATIC TDC control system. It is interesting especially for first-time users.</p> <p>System software This section communicates basic knowledge of the structure of a CPU's operating system and application programs. It should be used under the aspect of obtaining an overview of programming methodology and using this information as a basis for designing user programs.</p> <p>Configuring communication This section communicates basic knowledge of the communication possibilities and how to configure links to communication partners.</p>
<p>D7-SYS - STEP 7, configuring CFCs and SFCs http://support.automation.siemens.com/WW/view/de/8776786/0/en</p>	<p>Basic software This section explains the principles of use and functions of the STEP 7 automation software. Beginners obtain an overview of the procedures to follow when configuring, programming, and commissioning a station. While working with the basic software, you can directly rely on the Online Help system that offers support when it comes to detailed questions on using the software.</p> <p>CFC The CFC language (Continuous Function Chart) offers you the possibility of designing graphic interconnections for blocks. While working with the particular software, you can always consult the Online Help to get answers to detailed questions regarding the use of the editors/compiler.</p> <p>SFC Configuring sequential controls using SIMATIC S7 SFCs (Sequential Function Chart). You create the sequential chart in the SFC Editor based on various graphic resources and position the SFC elements of the chart according to defined rules.</p>
<p>SIMATIC TDC hardware http://support.automation.siemens.com/WW/view/de/8776697/0/en</p>	<p>These manuals form a reference for the complete hardware spectrum.</p>
<p>SIMATIC D7-SYS Selecting function blocks http://support.automation.siemens.com/WW/view/de/14952400/0/en</p>	<p>The Reference Manual provides you with an overview of all of the function blocks for the corresponding Technology and Drive Control components, i.e. SIMATIC TDC, FM 458-1 DP as well as the T400 and SIMADYN D systems, which are being discontinued.</p> <p>Section 1 This section describes the function blocks that can be configured in all target systems of SIMATIC D7-SYS.</p> <p>Section 2 This section describes the function blocks that can be configured only for SIMATIC TDC.</p> <p>Section 3 This section describes the function blocks that can be configured only for the FM 458-1 DP application module.</p> <p>Section 4 This section describes the function blocks that can be configured only for SIMADYN D and T400.</p>

Signpost

As first-time user, you should use the manual as follows:

- Read the initial sections before using the software so that you become familiar with the terminology and procedural principles.
- You can then go ahead and use the respective sections of the manual, for example, if you intend to run a specific task (e.g. loading programs).

If you have already gained some experience while running a small project, you can read individual sections of the manual in order to obtain information on specific topics.

Special notes

The objective of the user part of this manual is to provide information on basic procedures, but does not contain any detailed instructions with individual step sequences. For more information on the software dialogs and their handling, refer to the Online Help.

Recycling and disposal

The products can be recycled due to their low-pollutant content. Contact a certified electronic-waste disposal company to recycle and dispose of your old equipment in an environment-friendly manner.

Additional support

- You can find information on the technical support offer in the appendix (Page 305) to this documentation.
- You can find the offer for technical documentation for the individual SIMATIC products and systems on the Internet (<http://www.siemens.com/simatic-tech-doku-portal>).
- You can find the online catalog and online ordering system on the Internet (<https://mall.industry.siemens.com>).

Table of contents

	Preface	3
1	Just a few steps away from the first project	13
1.1	Requirements.....	13
1.1.1	Software and hardware.....	13
1.1.2	What to can expect	15
1.2	Creating a new project.....	16
1.3	Specifying the hardware	16
1.4	Creating CFC charts	17
1.4.1	Creating a new chart.....	17
1.4.2	Inserting, parameterizing and interconnecting function blocks.....	18
1.5	Testing, compiling, and downloading the project.....	21
1.5.1	Consistency check and compilation of the project.....	21
1.5.2	Downloading the user project to the SIMATIC TDC-CPU module	21
1.6	Checking the hardware configuration	23
1.7	Testing the user project	24
1.7.1	Shutting down the online connection	25
1.7.2	Creating a connection in online mode	25
1.7.3	Editing the parameterization in online mode	25
1.7.4	Inserting blocks in online mode	26
1.7.5	Deleting blocks in online mode	26
1.8	Results	27
1.9	Archiving projects.....	27
2	System software	29
2.1	Configuration.....	29
2.1.1	Configuring the hardware	31
2.1.1.1	Getting started: Selecting the hardware components.....	31
2.1.1.2	The second step: Selecting the hardware components.....	32
2.1.1.3	The third step: Verifying the configuration	33
2.1.2	Creating CFC charts	34
2.1.2.1	Getting started: Selecting the function blocks	34
2.1.2.2	The second step: Parameterizing and interconnecting the function blocks	35
2.1.2.3	The third step: Compiling the user program and loading it into the CPU	40
2.1.3	Operating states of a CPU module	42
2.1.4	Description and usage of signal transfer mechanisms	44
2.1.4.1	Data consistency.....	44
2.1.4.2	Data exchange within the same task of a CPU	44
2.1.4.3	Data exchange between different tasks of a CPU.....	45
2.1.4.4	Data exchange between cyclic tasks of several CPUs.....	46
2.1.4.5	Data exchange between interrupt tasks of several CPUs	47
2.1.4.6	Minimizing deadtimes	47

2.1.4.7	Processing sequence within a basic CPU clock cycle.....	48
2.1.4.8	Interconnection changes and limited number of interconnections.....	49
2.1.5	Significance and application options of the process image	50
2.1.5.1	Implementing the process image	51
2.1.5.2	Process image for cyclic tasks.....	52
2.1.5.3	Process image for interrupt tasks	53
2.1.6	Significance and application of CPU synchronization.....	54
2.1.6.1	Time synchronization	54
2.1.6.2	Synchronization of the own basic clock cycle to the basic clock cycle of a master CPU	55
2.1.6.3	Synchronizing the own basic clock cycle to an interrupt task of a master CPU	55
2.1.6.4	Synchronizing the own interrupt tasks to interrupt tasks of a master CPU	55
2.1.6.5	Synchronizing multiple SIMATIC TDC stations	55
2.1.6.6	Response to synchronization failure	55
2.1.6.7	Configuring synchronization of the CPU basic clock cycle	56
2.1.6.8	Configuring interrupt task synchronization.....	57
2.1.7	Significance of the processor utilization.....	58
2.1.7.1	Determining the approximate processor utilization.....	58
2.1.7.2	Precise calculation of processor load	59
2.1.7.3	Illustration of the mode of operation of the Task Manager	60
2.1.7.4	Eliminating cycle errors.....	62
2.1.7.5	Error displays on task overflows	62
2.1.8	Technical specifications of the operating system	63
2.1.8.1	Performance features	63
2.1.8.2	Basic functions of the operating system	65
2.1.8.3	Service utility	70
2.2	Function description and user instructions	72
2.2.1	Fatal system error "H"	72
2.3	System chart @SIMD	75
3	Configuring communication.....	79
3.1	Introduction	79
3.1.1	Communication basics.....	79
3.1.2	Overview of communication utilities.....	80
3.1.3	Overview of couplings.....	81
3.1.4	Communication block I/O.....	85
3.1.4.1	Initialization connection CTS.....	85
3.1.4.2	Address connections AT, AR, and US.....	87
3.1.4.3	Transmission mode, MOD connection.....	89
3.1.4.4	Firmware status, ECL, ECO connection	94
3.1.4.5	Status display, output YTS.....	94
3.1.5	Function principle of couplings.....	95
3.1.5.1	Central coupling blocks.....	96
3.1.5.2	Transmitters and receivers	97
3.1.5.3	Compatible user data structures.....	99
3.1.5.4	Number of coupling modules in a rack	101
3.1.5.5	Reorganizing data interfaces	101
3.2	Local couplings in the rack.....	102
3.2.1	Local CPU coupling	102
3.2.2	Direct CPU-CPU coupling	103
3.2.3	Buffer memory coupling	105
3.2.4	Application notes.....	106

3.3	CP52M0 rack coupling	107
3.3.1	Areas of application	107
3.3.2	Power on/off response	108
3.3.3	Synchronization and trigger options	109
3.3.4	Configuration.....	109
3.3.4.1	Configuration in HW Config	110
3.3.4.2	Configuration in CFC	113
3.3.5	Performance data	113
3.4	CP53M0 rack coupling	114
3.4.1	Hardware installation	117
3.4.2	Scope of performance.....	117
3.4.3	Response to the "shutdown" of a coupling partner.....	118
3.4.4	Response to "power on" of the master rack	119
3.4.5	Restart capability	121
3.4.6	Configuration.....	121
3.4.7	Restrictions	122
3.5	TCP/IP coupling (CPU555; CP51M1).....	123
3.5.1	Comparison of TCP/IP with UDP	124
3.5.2	Configuration model.....	126
3.5.3	Configuration steps	126
3.5.3.1	Configuration in HW Config	126
3.5.3.2	Configuration with CFC.....	127
3.5.4	Application notes.....	130
3.5.5	Communication via WinCC.....	132
3.5.6	Central service	132
3.5.7	Time synchronization	132
3.5.8	Migration from CP5100 to CP51M1	132
3.6	PROFIBUS DP coupling (CP50M1).....	133
3.6.1	General basics	133
3.6.2	Configuration.....	134
3.6.3	Constant bus cycle time.....	137
3.6.4	SYNC/FREEZE commands	137
3.6.4.1	SYNC/FREEZE configuration variants	138
3.6.5	Commissioning/diagnostics	142
3.6.5.1	Diagnostics function block	142
3.6.5.2	Error class (ECL) and error code (ECO).....	144
3.7	MPI coupling	145
3.8	PNIO communication	146
3.8.1	Overview	146
3.8.2	Configuration model.....	148
3.8.3	Configuring SIMATIC TDC	149
3.8.4	Use isochronous I/Os.....	153
3.9	Table function	156
3.9.1	Introduction	156
3.9.1.1	"Manual mode" overview	157
3.9.1.2	"Automatic mode: Communication" overview	157
3.9.1.3	"Automatic mode: Memory card" overview	159
3.9.1.4	Function block WR_TAB.....	159

3.9.2	Manual mode	162
3.9.3	Automatic mode: Communication	164
3.9.3.1	Application with S7 controller and SIMATIC FM 458 application module.....	164
3.9.3.2	Configuration for the S7 controller and FM 458 application module.....	166
3.9.3.3	Inserting tabular values in the data block	167
3.9.3.4	Loading additional tabular values to a DB	181
3.9.3.5	Structure of the data frame for TCP/IP or DUST1 connections	182
3.9.4	Automatic mode: Memory card	183
3.9.4.1	Creating a table file in csv format.....	183
3.9.4.2	Working with the D7-SYS additionalComponentBuilder	186
3.9.4.3	Loading	190
3.9.4.4	Configuring the function blocks.....	192
3.10	Communication utility alarm logging	194
3.10.1	Logging logic of the alarm logging blocks	195
3.10.2	Example of an alarm logging configuration.....	196
3.10.3	Output formats of the alarm evaluation block MSI.....	199
3.10.3.1	Structure of an error or alarm message	199
3.10.3.2	Overview of alarm formats	200
3.10.3.3	Structure of an overflow alarm	202
3.10.3.4	Structure of a communication error message	202
3.10.3.5	Structure of a system alarm	203
3.10.3.6	Detailed description of the alarm formats of function block MSI.....	204
3.10.3.7	Output format of the message evaluation block MSIPRI	209
3.11	Communication utility process data	211
3.11.1	Receive and transmit blocks	211
3.11.1.1	Virtual connections.....	211
3.11.1.2	Connections of the CRV, CTV blocks.....	215
3.11.2	Channel marshalling blocks CCC4 and CDC4	216
3.11.2.1	Group block CCC4.....	216
3.11.2.2	Distribution block CDC4.....	218
3.11.2.3	Compatible user data structure.....	218
3.11.3	Diagnostics outputs.....	219
3.11.3.1	Error cause.....	219
3.11.3.2	Channel assignment	221
3.11.3.3	Channel states	221
3.11.4	Getting started with "pointer-based communication blocks".....	222
3.11.4.1	Function principles	222
3.11.4.2	Applications.....	223
3.11.4.3	Features of pointer-based communication	224
3.11.4.4	Corresponding function blocks.....	225
3.11.4.5	Pointer interface	225
3.11.4.6	Notes on configuration	226
3.11.4.7	Examples based on CFC screenshots	227
3.12	Communication utility service	232
3.12.1	SER function block.....	234
3.12.2	System load, response times.....	235
3.13	Communication utility time synchronization	236

3.14	WinCC connection to SIMATIC TDC via standard channel (SIMATIC S7 Protocol Suite.CHN).....	239
3.14.1	Coupling over TCP/IP with "OCM" functions	241
3.14.1.1	Configuring the coupling-relevant TDC hardware	241
3.14.1.2	CFC configuration.....	242
3.14.1.3	WinCC configuration.....	252
3.14.2	"S7DB" configuration variant	258
3.14.3	MPI and PROFIBUS DP coupling variants	259
3.14.3.1	Hardware configuration.....	260
3.14.3.2	CFC configuration.....	265
3.14.3.3	WinCC configuration.....	266
3.14.4	Configuration using the "D7-SYS-OS Engineering Tool"	268
3.15	Communication with WinCC (TCP/IP)	278
3.15.1	Requirements.....	278
3.15.2	Process tags	279
3.15.3	Bit messaging	281
3.15.4	SIMATIC TDC messages.....	282
3.15.5	Address book generation in CFC editor.....	284
3.15.6	Communication setup SIMATIC TDC <-> WinCC	284
3.16	Communication utility Trace	285
3.16.1	Simple Trace	285
3.16.1.1	Mode of operation of @TCP	285
3.16.1.2	Mode of operation of the acquisition blocks	288
3.16.1.3	Mode of operation of header block TRHI.....	289
3.16.1.4	Simple Trace configuration	289
3.16.1.5	Response frames.....	292
3.17	TDC-OPC server connection	297
3.17.1	Overview	297
3.17.2	Configuration model.....	298
3.17.3	Configuring an OPC server connection	299
3.18	Connections across network boundaries (routing)	304
A	Service & Support.....	305
A.1	Service & Support.....	305
	Index.....	309

Just a few steps away from the first project

1.1 Requirements

Introduction

This Getting Started is intended for newcomers and outlines the basic procedures for creating projects.

For more information on the dialogs of the development software and their processing, refer to the corresponding Online Help.

1.1.1 Software and hardware

Software

The three software packages

- STEP 7
- CFC
- D7-SYS

must be installed precisely in this order on your PG/PC with Windows. Authorization is required for STEP 7 and CFC.

Note

Current notes on installation and user instructions are available in the respective "readme" files. Observe the version dependencies!

When installing STEP 7, you will be prompted for the online interface. However, for SIMATIC TDC, nothing has to be selected and installed. ("Close" window and close the following window with "OK".)

1.1 Requirements

Hardware

You need the following hardware components for the "My First Project" project template:

Table 1- 1 Module list for the "My First Project" template

Component	Function	Article number
UR6021 rack with power supply <i>21 slots, 64-bit bus, fan, 115/230 VAC</i>	... if the rack is for a SIMATIC TDC station. ... used for mechanically mounting the modules and supplying them with power.	6DD1682-0CH3
CPU module CPU555 <i>(in slot 1)</i> <i>64-bit, 2 GHz, 4 GB SD-RAM, 3 PROFINET interfaces</i>	... runs the user program. ... exchanges data with other modules via the backplane PCB of the rack. ... communicates with a PG/PC via the PROFINET interface.	6DD1600-0BB0
Program memory module <i>2 MB user program memory</i>	... stores the operating system, user program and online changes.	6ES7953-8LL31-0AA0
Ethernet cable <i>RJ45/RJ45, 2 m</i>	... connects the CPU module to the PG/PC.	6XV1870-3QH20
SM500 signal module <i>(in slot 2)</i> <i>16 BQ, 16 BI, 8AI, 4AI integrating, 8AQ, 4 pulse encoder inputs, 4 absolute encoder inputs</i>	... provides analog and digital inputs and outputs, as well as incremental and absolute encoder I/Os.	6DD1640-0AH0
Interface cable SC 62 <i>Length: 2 m</i>	... connects the inputs/outputs of the SM500 module with up to 5 SBxx or SU12 interface modules.	6DD1684-0GC0
Interface module SB10 <i>2 x 8 screw terminals, LED displays</i>	... allows you to test the user program during commissioning and operation, because the states of the digital outputs are displayed with LEDs.	6DD1681-0AE2

Note

You can find the technical specifications in the "SIMATIC TDC Hardware" system manual.

You can find ordering information on the Internet (<https://mall.industry.siemens.com>).

1.1.2 What to can expect

From the task to the first project

The "My First Project" template guides you step-by-step on the way to an executable project.

1. Task analysis

This analysis helps you to select the function blocks, for reference potentials, and hardware you need:

2. Specifying the hardware

You are going to use this hardware information in STEP7 in order to enter the modules and specify their properties.

3. Configuring and compiling

In the CFC, create the configuration using the function blocks and then compile this data. Install the hardware after having completed all checks.

4. Testing the configuration

You can now run the program on the SIMATIC TDC modules and test or edit it in online mode.

5. Archiving projects

You may apply this procedure to you own applications.

The task

The task consists of two parts:

1. A **sawtooth generator** with fixed frequency outputs its value via D/A converter.
2. A **running light** with eight channels.

To start off with, define the individual functions for the corresponding task elements and specify the necessary hardware:

1. Sawtooth generator

A sawtooth signal is generated by an integrator that resets as soon as an upper limit has been exceeded. The integrator value is returned via analog output.


2. Running light

Eight comparators compare the sawtooth value with constants. The results are returned at digital outputs and control the LEDs on the interface module.

The running light has the following phases:

- All LEDs are dark.
- The LEDs are lit and then toggled to dark state again so that only one LED is lit at any given time.

1.2 Creating a new project

Step	Procedure	Result
1	<p>Double-click the following "STEP 7" icon</p>  <p>(if the STEP 7 Assistant starts, cancel this.)</p>	SIMATIC Manager opens.
2	<p>Select File > New.</p> <p>Enter "My First Project" in the Project dialog.</p> <p>Select the storage location (path) "Drive:\Siemens\Step7\S7proj".</p> <p>Click OK.</p>	Your new project is displayed.
3	<p>Select Insert > Station > SIMATIC TDC station.</p>	The "SIMATIC TDC station" hardware object is inserted.

1.3 Specifying the hardware

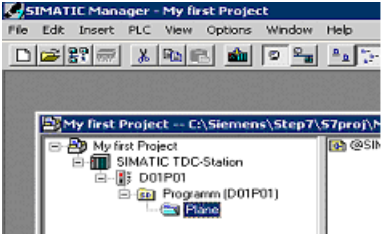
The SIMATIC TDC rack structure is entered in STEP 7 (HW Config).

Step	Procedure	Result
4	<p>Select the "SIMATIC TDC station" hardware object and then select Edit > Open object.</p>	HW Config opens.
5	<p>You can open the hardware catalog with View > Catalog.</p>	The Hardware Catalog opens and lists all available module families.
6	<p>Select the UR6021 from the SIMATIC TDC product series and the Rack catalog and drag-and-drop it to the (upper) window.</p>	The rack is displayed with 21 slots.
7	<p>Position the following modules in successive order:</p> <ul style="list-style-type: none"> • CPU Modules > CPU555 in slot 1 and set the parameters for the "Ethernet interface PN-IO". • Signal Modules > SM500 in slot 2 • Racks > SR51 in slots 3 to 21 	The rack is now equipped.
8	<p>Select Edit > Object properties to open the properties dialog of the CPU555 CPU module.</p>	The CPU555 dialog opens and displays general module information and the configuration tabs General, TSave, Basic clock cycle, Cyclic tasks, Interrupt tasks, Isochronous mode and Stop.
9	<p>Select the basic sampling time T0 in the Basic clock cycle tab (in this case: 1 ms).</p> <p>Click on the Cyclic tasks tab and set the sampling time T1 to 2 ms and T2 to 4 ms.</p> <p>Click OK.</p>	The necessary sampling times have been entered. The Properties dialog is closed.

Step	Procedure	Result
10	Select Edit > Object properties to open the Properties dialog of the SM500 signal module.	The SM500 dialog opens and displays general module information and the tab for setting addresses.
11	In the Addresses tab, click on the Preset button. Click OK .	All of the addresses are assigned symbolic names for subsequent use in CFCs.
12	Select Station > Check consistency to check your hardware setup.	If fault/error-free, continue with Step 13, otherwise check the hardware configuration.
13	Compile your hardware configuration with Station > Save and Compile .	The hardware configuration has been successfully completed.

1.4 Creating CFC charts

1.4.1 Creating a new chart

Step	Procedure	Result
14	Change to SIMATIC Manager and expand the project tree up to the Charts object. Click the charts you want to select.	
15	Create two new CFC charts with Insert > S7 software > CFC .	The new CFC1 and CFC2 chart objects are displayed in the right pane of the project window.
16	Select chart CFC1 in the project window and open the properties dialog box with Edit > Object properties . Enter the name "sawtooth generator". Click OK .	You are returned to the properties dialog the CFC, which is now closed.
17	Repeat step 16 with the CFC2 chart and rename it to "Running lights".	The charts appear in the project window under their new name.

1.4.2 Inserting, parameterizing and interconnecting function blocks

Step	Procedure	Result
18	Select the "sawtooth generator" chart and open the "CFC Editor with Edit > Open object .	The CFC Editor opens and displays the work area (1 sheet) and the block catalog. (Catalog missing? Select View > Catalog) (>1 Sheet? Select View > Sheet view)
19	Open the Control block family and drag-and-drop the INT (integrator) function block to the work area.	The block is now positioned on the sheet and has the ID for runtime in cyclic task T1.
20	Open the properties dialog box of function block INT with Edit > Object properties .	The INT dialog that opens displays general block information and the for reference potentials settings tab.
21	In the General tab, rename the object to "sawtooth".	
22	In the I/Os tab, enter the values for the block inputs, e.g. <ul style="list-style-type: none"> • X = 1 • LU = 11250 • TI = 5 ms Click OK .	The properties dialog is closed and the function block inputs have now been assigned their values.
23	Click on output QU and then on input S.	Output QU (upper limit) is now fed back to input S (set).
24	Select the DAC (analog output) from the ON/OFF block family and place it next to function block INT. Open the dialog box using Edit > Object properties and rename the output to "analog output". In the for reference potentials tab enter the following values, for example: <ul style="list-style-type: none"> • DM= 0 • OFF= 0 • SF= 1E6 Click OK . Select the AD (hardware address) connection, open the object interconnection dialog with Insert > Connect to operand and then open the selection window. Select the first entry and click OK	The block inputs are parameterized. You assigned the hardware address to the first analog output channel.
25	In the "sawtooth" block, click on output Y and then on input X in the "analog output" block.	This interconnects the sawtooth generator with the analog output.

All changes made to the CFC chart are saved immediately.

Complete the second part of the task (running lights) based on the same procedure (as of step 18).

Change to SIMATIC Manager, open the CFC chart "Running lights", insert the function blocks into the CFC chart, parameterize, and interconnect these.

Consult the following figures for all necessary information (number of blocks, block types, and block parameters). Arrange the first and all other function blocks in cyclic task T2 using the **Edit > Run sequence** command. Change to the CFC window (**Window > ...**) to interconnect the "sawtooth" block to the comparators.

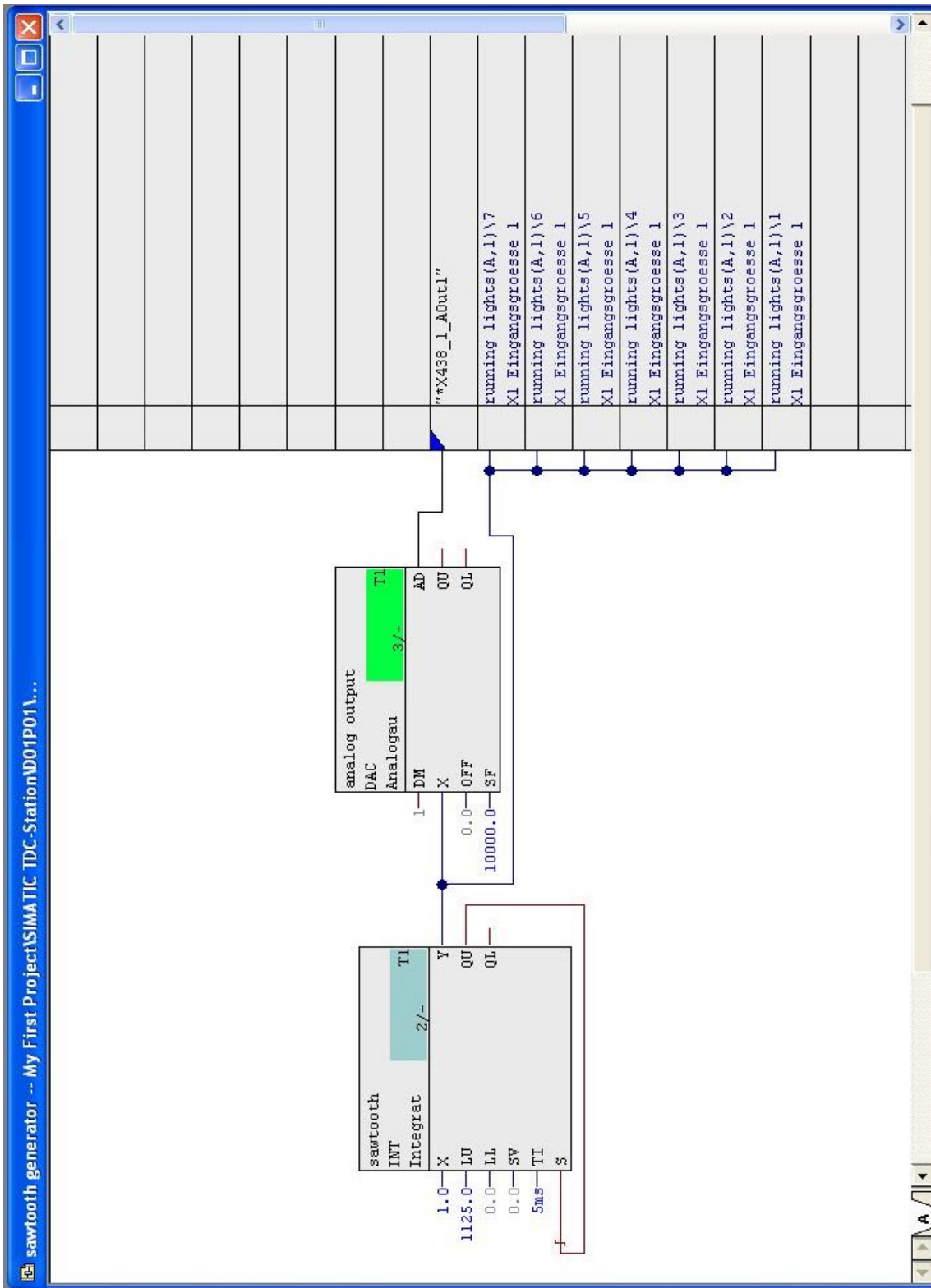


Figure 1-1 "Sawtooth generator" chart

1.4 Creating CFC charts

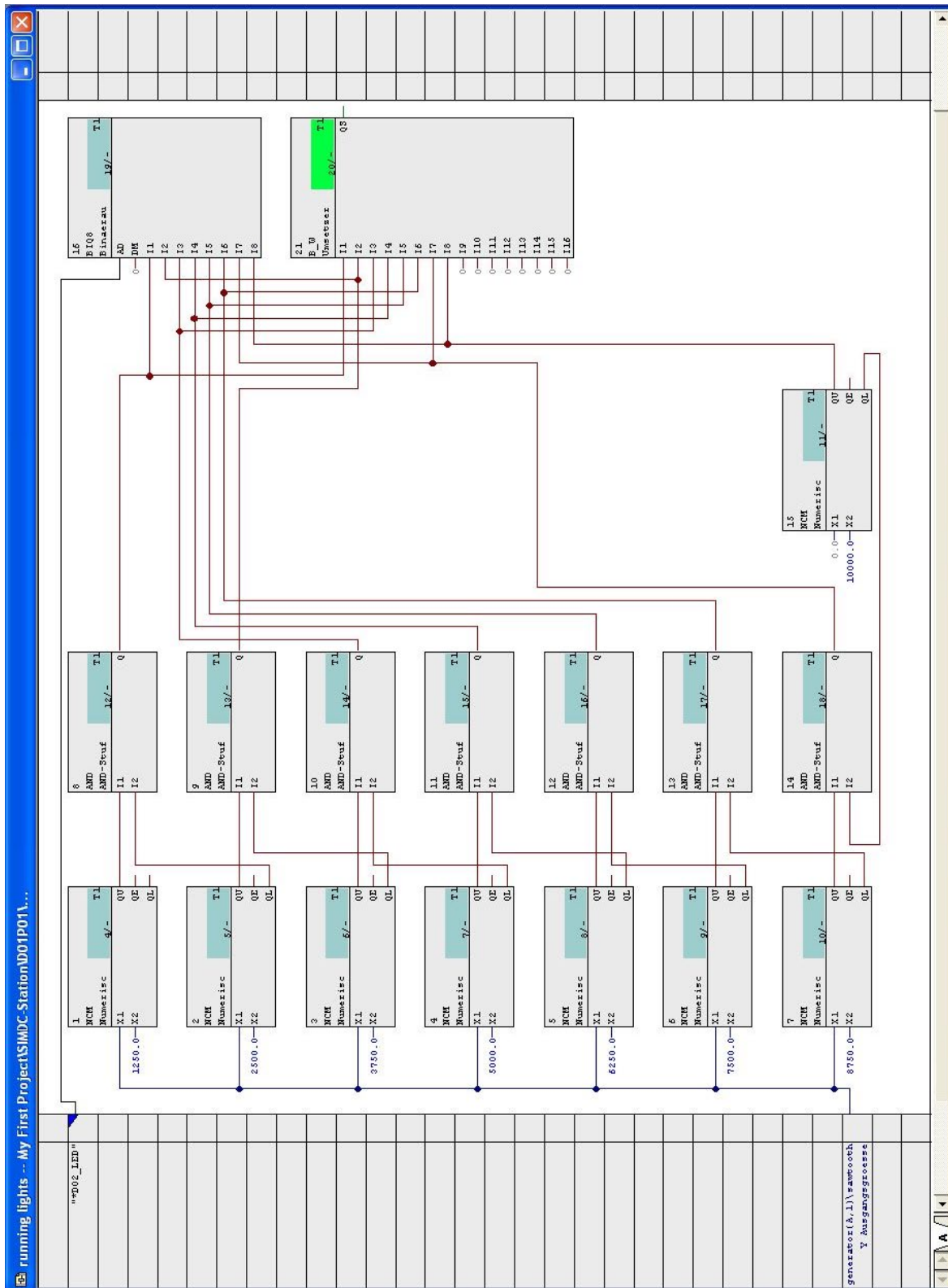


Figure 1-2 "Running lights" chart

1.5 Testing, compiling, and downloading the project

1.5.1 Consistency check and compilation of the project

Step	Procedure	Result
26	Select Chart > Check consistency > Charts as program and then click OK to check the consistency of your project. Acknowledge the dialog window, or take a closer look at the error messages using Details .	A dialog window displays the result.
27	On successful completion of the consistency check, select Chart > Compile > Charts as program and then click OK to compile the project. Acknowledge the dialog window, or take a closer look at the error messages using Details .	A dialog window displays the result. You have created your first user project.

1.5.2 Downloading the user project to the SIMATIC TDC-CPU module

Introduction

SIMATIC TDC offers the following download options:

- Online
- Offline

Loading offline

If you do not have a connection from your PC/PG to the SIMATIC TDC station, use the option of loading the project data to a memory module.

Step	Procedure	Result
28	Select Target system > Download .	A dialog window opens and displays the options.
29	Select "User program" and "Offline". Insert the memory module into the appropriate slot of the PG/PC. Click OK to start loading.	A progress bar indicates the progress of the operation for writing the system and user program data to the memory module.
30	Insert the memory module into the SIMATIC TDC station and restart the station.	Your user program is started.

Online downloads

You have set up a connection from your PC/PG to SIMATIC TDC station and are able to write the project data directly to the program memory module in the CPU module.

Step	Procedure	Result
28	Verify that your SIMATIC TDC station (hardware) is properly configured and wired.	Observe the installation notes and connection options provided in the corresponding hardware documentation for the various hardware components!
29	Insert the memory module into the CPU module and start the SIMATIC TDC station.	A flashing "0" is shown on the CPU module display.
30	In SIMATIC-Manager, select the Tools > Set PG/PC interface... menu command to install the interface between the SIMATIC TDC station and the PC.	The "Install/uninstall interfaces" dialog opens and displays list of interfaces.
31	In the dialog window, select the TCP/IP interface used and install this protocol with Install > Confirm with "Yes" and Close the dialog window. Select the interface used and confirm with "OK".	In the next dialog, decide whether or not to go online immediately by selecting "Yes" or "No". The "Set PG interface" dialog window opens in which you can select the access route "TCP/IP".
32	Select Target system > Download.	A dialog window opens and displays the options.
33	Select "System and user program", "Online (TCP/IP)", and CPU memory reset when initially loading the user program. Note: For repeated downloads of the user program, you can select only the "User program" and dispense with a "CPU memory reset". Start the "download"	A progress bar indicates the progress of the operation for writing the system and user program data to the memory module. On completion of the download, the "STOP" status is displayed in the "Operating status" dialog window.
34	Start the SIMATIC TDC station with "Restart" and then select "Close".	Your user program is started and the "RUN" status is displayed in the "Operating status" dialog window.

Note

Online downloads

- CPU551: The speed of the download can be significantly reduced with a heavy load on the CPU.
- CPU555: The speed of a local download is limited by the available bandwidth on the PROFINET IO and is independent of the load on the CPU.

1.6 Checking the hardware configuration

You can read out the states of modules with diagnostics capability and check them for freedom from errors with the HW Config online diagnostics. You can test the accessibility of the modules and devices in PROFINET and detect differences in the online/offline configuration.

Follow these steps to check the configuration:

1. Access the SIMATIC TDC station with an online connection.
2. Select the menu command "Station > Open online" in HW Config.
3. Select the module from which you want to read diagnostics data.
4. Select "Target system > Module state".

Note

To be able to use the HW Config online diagnostics, the modules involved must have at least the following versions:

- CPU555: V1.1 (only some of the information is displayed for V1.0)
 - CP51M1: V1.1.5
 - CP50M1: HW 5
-

1.7 Testing the user project

Introduction

In test mode, you can

- monitor the values of the block for reference potentials, edit the values of block inputs,
- create and delete connections, and
- insert and delete blocks.

The values which are registered for testing are displayed on a yellow background. You can easily monitor the response by changing block input parameters.

Before you start the test, check whether the following conditions are fulfilled:

- A connection is set up between the PG/PC and your SIMATIC TDC station.
- You have downloaded the actual project to the memory module on the CPU module.
- The corresponding CFC chart (e.g. "running lights") is open.

Step	Procedure	Result
35	Select Target system > Compare menu command to display the "Compare" dialog.	The CPU name and the time stamp of the most recent compilation and the result of the comparison between the actual configuration and the current CPU program are displayed. If consistent, the result "The configuration and CPU program match" is returned. The result of this check shows that communication between the PG/PC and the SIMATIC TDC station is possible.
36	Select Test > Test settings Enter the screen refresh period in tenths of a second. Confirm your changes with "Update."	In test mode, the on-screen for reference potentials values are updated at cyclic intervals based on the selected refresh period. A warning is output if the time slice for computing is insufficient for the refresh period. The control system always takes higher priority
37	Before switching to the test mode, change the test mode setting from "Process mode" to "Laboratory mode" with Test > Laboratory mode . Note: A monitoring connection is not set by default for the "Process mode". In this test mode, you must select and explicitly register the corresponding blocks for monitoring.	This means that all block for reference potentials are automatically enabled for "monitoring" (values on yellow background).
38	Select Test > Test mode	The "Test: RUN (laboratory)" is displayed on a green background in the status bar. In the test mode, you can monitor and modify the dynamic response (online).

1.7.1 Shutting down the online connection

Procedure

In the CFC chart, use the mouse to select the block for reference potentials that you want to disconnect. Then remove the connection with **Edit > Delete**.

Result

The connector between the for reference potentials is cleared and the last value that was transferred via this connection is displayed as parameter value at the for reference potentials.

Note

It is not possible to create or delete connections to global operands in online mode.

1.7.2 Creating a connection in online mode

Procedure

In the CFC, use the mouse to select the block for reference potentials that is to be used as connection source.

While keeping the SHIFT key pressed, select the block for reference potentials that is to be used as connection target.

Result

The connector between the selected for reference potentials is generated and the actual parameter value that is currently transferred appears at the output.

1.7.3 Editing the parameterization in online mode

Procedure

Select the relevant block input with double-click to edit its parameter value. The "for reference potentials properties" dialog in which you can edit the value is opened.

Result

You can immediately see the effect of the change in the CFC Chart

1.7.4 Inserting blocks in online mode

Procedure

Use the **View > Catalog** command to open the Block Catalog. Open the block family and drag-and-drop the selected function block to the work area.

Note

Not all of the function blocks can be inserted in online mode. For help on the block, refer to "Configuration data" in the Online Help.

1.7.5 Deleting blocks in online mode

Procedure

Select the function block and remove it using the **Edit > Delete** command.

1.8 Results

You have now been introduced to the most fundamental activities in CFC configuration. You now know how to create a project using SIMATIC Manager, how to create a CFC Chart, and import function blocks from a library. You have interconnected and parameterized the function blocks. You have created an executable program and downloaded it to the CPU. You also learnt how to monitor and modify dynamic response in test mode.

You can view the results for the "My First Project" example in **online** process mode if you have installed and wired the necessary hardware for the SIMATIC TDC station (refer to Table 1-1, section 1.1.2).

Sawtooth generator

You first have to connect an oscilloscope to the SIMATIC TDC station if you want to view the sawtooth signal. The following table lists the pin assignment of output connector X1 of the SM500 signal module.

The output voltage has a range from -10 V to +10 V.

Table 1-2 Extract from the connector X1 pin assignment on SM500

Pin	Function	Output
1	Analog output 1+	Sawtooth
2	Analog output 1 -	

Running light

You can observe the running light function on the LED display of interface module SB10.

1.9 Archiving projects

Step	Procedure	Result
44	Select File > Archive in SIMATIC Manager.	The "Archiving" dialog is displayed.
45	In the "Archiving" dialog, select the "My First Project" user project. Click OK .	The "Archiving - select archive" dialog is displayed. The default file "My_first.zip" has already been entered along with storage path.
46	In the "Archiving - select archive" dialog, rename the file and/or the path and then click "Save"	The project is now saved to the selected path and file names as Zip file.

Note

You may always select **File > Retrieve** to restore this project version from the archive.

System software

2.1 Configuration

This section is intended to guide and support you in configuration. It explains the general conditions for configuration SIMATIC TDC hardware and software.

It is presumed that readers have sufficient knowledge of Windows, as well as of the operation of SIMATIC Manager, HW Config and CFC Editor, which is why these topics will not be covered. The configuration instructions are explained based on pictures and graphic images. These illustrations are intended to highlight specific features and do not necessarily represent the CFC windows. As this manual does not cover the hardware (e.g. CPUs, memory modules, cables, etc.), even if hardware designations are used in the example configurations, you should rather consult the corresponding "SIMATIC TDC Hardware" manual.

This manual is divided into the following sections:

- General description
- Configuring the hardware
- Creating CFCs
- Operating states of a CPU module
- Example of a CPU module configuration
- Using signal transfer mechanisms
- Significance and application options of the process image
- Significance and areas of application of CPU synchronization
- Significance of the processor utilization

The information provided as of section "General description" and up to section "Creating CFCs" is sufficient for you to implement most of the applications. In-depth information regarding the special system properties of SIMATIC TDC is available in the sections that follow.

Configuration tools

In practice, project engineers can rely on their sound knowledge when selecting hardware modules from a product range and when creating function charts or block diagrams in order to realize the desired technological functions. SIMATIC TDC supports these activities using the HW Config (configuration tool used to define the hardware configuration of SIMATIC TDC stations) and CFC (block technology using numerous standard function blocks) applications.

Configuration steps

SIMATIC TDC is configured by the following configuration steps:

1. Creation of the hardware configuration
2. Creation of the CFCs.

Name assignment

General rules concerning the assignment of names for the configuration of SIMATIC TDC:

- Station names
 - max. 24 characters
- Modules
 - maximum length of 6 characters.

Table 2- 1 Nomenclature for naming modules

Character strings	Valid characters	Example
First character	Alphanumerical and special characters	A-Z, @
Second character	Alphanumeric and special characters	A-Z, 0-9, _ , or @ if the first character is @
Additional characters	Alphanumeric and special characters	A-Z, 0-9, _

- Chart or function block names
 - The maximum length of 24 characters may not be exceeded in a string consisting of both names.

Table 2- 2 Nomenclature for naming charts and function blocks

Name	Max. length	Valid characters	Invalid characters
Chart	22		*, _, ?, <, >,
Function block	16		"

- Comments be up to
 - 255 characters for modules
 - 255 characters for charts
 - 80 characters for function blocks and parameters

in length.

- I/Os with special functions have the following prefixes:
 - Dollar symbol "\$" (interconnection of signals between CPUs)
 - Star symbol "*" (symbolic hardware addresses)
 - Or the exclamation mark "!" (virtual addressing)

HW Config or CFC automatically attaches these prefix characters. Function block names must be unique on a CPU. Compliance with naming conventions is checked during input.

Libraries

Hardware modules and function block types are assembled in libraries. You can call the function blocks you need from the libraries using HW Config or the CFC editor.

It is possible to use several function block libraries for each CPU. The default "FBSLIB" standard function block library provides more than 200 function blocks with appropriate functionality for most applications. Additional libraries can be "imported" for the respective CPU. The libraries are stored in the directory "...\\tdc (SIMATIC TDC)".

2.1.1 Configuring the hardware

Configuring a SIMATIC TDC station

HW Config is used to create the hardware configuration of SIMATIC TDC stations. A SIMATIC TDC station consists of a rack with up to 10 CPU551 or 8 CPU555 and other hardware modules (see SIMATIC TDC hardware (<http://support.automation.siemens.com/WW/view/de/8776697/0/en>)). It is possible to couple several stations. The modules to be configured can be selected from the product range listed in the hardware catalog of HW Config. This catalog contains a selection of racks, CPUs, for reference potentials modules, coupling modules, etc.

HW Config defines the hardware configuration of the system base on:

- The rack used, along with the definition of the bus structure (bus termination, daisy chain)
- The configured hardware modules inserted in this rack
- The definition of hardware-relevant information such as tasks, or synchronization.

2.1.1.1 Getting started: Selecting the hardware components

The following modules are available in the hardware catalog of HW Config:

Short overview of the hardware

Table 2- 3 Hardware components

Hardware	Description
Rack	Various Types, bus features, ventilation/fan etc. (see SIMATIC TDC hardware (http://support.automation.siemens.com/WW/view/de/8776697/0/en))
Input/output modules	I/O modules for the input/output of process signals (analog/binary I/O, speed sensing functions, etc.)
Expansion modules	I/O modules for the input/output of process signals. These are used to achieve higher data rates by bypassing the backplane bus and are directly interconnected with a CPU module.
Communication modules	Modules that provide communication services
Buffer memory module	Modules for handling data exchange between several CPUs.
CPU modules	Modules on which the configured control program is executed. You may insert up to two expansion modules next to a CPU.
Special modules	Modules with special functions.
Slot covers	Slot covers for empty slots prevent the ingress of dirt and also serve as EMC measure
Submodules	Module that is inserted in or on a module, e.g. a memory module for a CPU, or an interface module for a communication module
Technology components	Subracks as well as modules for current converters

A module is configured for each rack slot, possibly with submodule using HW Config. This provides a precise image of the rack for configuration. Each module called is assigned a default name that you may change in accordance with naming conventions. You should always protect unused slots with slot covers.

You can find **more information** on the individual modules and submodules in the system manual SIMATIC TDC hardware (<http://support.automation.siemens.com/WW/view/de/8776697/0/en>).

2.1.1.2 The second step: Selecting the hardware components

The modules you select must be configured using HW Config. This includes the following settings:

- Sampling times of the cyclic tasks
- Synchronization of the cyclic or interrupt-driven tasks of several CPUs of a station
- Hardware interrupts and comments
- IP address (CPU555)

For his purpose, HW Config provides different parameter assignment dialogs.

Parameter assignment dialogs in HW Config

The defaults may be changed in the dialog of the respective modules. For instance, the configuration dialog for CPU modules includes the "Cyclic tasks" specification. This allows you to edit the sampling times of the five cyclic tasks.

Naming scheme

At least one rack, including all modules and submodules it contains, must be configured in HW Config. When creating a module, you are proposed a default module name. This proposed name may be overwritten in compliance with the maximum name length (max. 6 characters) and valid characters (A-Z,0-9,_,@) (see Configuration (Page 29)). It is recommended to select the names based on the scheme for system components shown in the following table:

Table 2- 4 Naming scheme for the hardware configuration in HW Config

Hardware	Logical name	Identifier	Significance
Rack	An00	n	Subrack number, starting at 1
CPU	Dxy_Pn	xy n	Slot number CPU number
PN controller (CPU555)	PN-IO-<Rack>-<CPU> Port x	x	x = number of the port
Submodule (CPU551)	Dxyj	xy j	xy = slot number Submodule number
Buffer memory module	Dxy__A	xy	xy = slot number
Rack coupling	Dxy__B	xy	xy = slot number
Serial couplings	Dxy__C	xy	xy = slot number
Other modules	Dxy	xy	xy = slot number

Slot number definition

The slot number of a module specifies the number of the slot in the rack on which the respective module has been configured. @SIMD cannot be deleted and is automatically configured with the slowest sampling time (T5).

All submodules of a module are numbered consecutively, starting at 1. The submodule at the top of the table is assigned number 1.

The recommended CPU rack name has a length of six characters. The logical processor number (in the rack, from left to right) is displayed during operation on the seven-segment display of the CPU module, , independent of the assigned name.

Note

The configured module names must be unique throughout the station.

The various tasks of a CPU

The configured function blocks are processed by means of

- 5 cyclic tasks and/or
- 8 interrupt tasks.

System chart

The system chart in which you configured the behavior, e.g. of the seven-segment display or acknowledge button, is managed in a new SIMATIC TDC program and may not be deleted. The sampling time of the system chart is set to a factory default of approx. 128 ms.

2.1.1.3 The third step: Verifying the configuration

On completion of the hardware configuration, perform a station-wide consistency check to verify the configured data. The error-free status of the entire hardware configuration is verified in HW Config. Incompleteness or errors found in the configuration data are displayed so that you can eliminate these (refer to section "Example of a CPU module configuration").

2.1.2 Creating CFC charts

Description of the CFC editor

A CFC chart (Continuous Function Chart) is created using the CFC editor. CFC editor is an engineering tool that is used to describe continuous processes by means of graphic interconnection of complex functions on the basis of individual function blocks. The CFC therefore contains the graphic realization of a technological task on the basis of the interconnection and parameterization of function blocks. This solution is oriented on block diagrams and facilitates program engineering.

CFC chart structure

A CFC consists of several CFC charts with 6 sheets. Each sheet may contain a different number of different function blocks. The number of blocks is only limited by the possible graphic layout. The overview of the CFC editor displays all 6 sheets of a chart, while the sheet view enables a detailed view of individual sheets. The function blocks that can be called in the CFC editor are split into function block classes that contain the associated functionality, e.g. logic blocks, or arithmetic blocks. Each function block class in turn contains a number of different function block types.

The CFC editor defines the technological configuration by means of:

- Selection, interconnection, and parameterization of the configured function blocks
- Specification of the runtime characteristics of the function blocks
- Generation of executable code for programming the CPU memory modules.

2.1.2.1 Getting started: Selecting the function blocks

The FBSLIB standard library provides different function block classes. You can call the individual function blocks in the CFC editor and place these in the chart sheets. You may always delete, move, and copy individual blocks or block groups.

Additional information on function blocks is available in the programming manual "Selecting D7-SYS function blocks

(<http://support.automation.siemens.com/WW/view/de/14952400/0/en>)".

2.1.2.2 The second step: Parameterizing and interconnecting the function blocks

The selected function blocks must be interconnected and parameterized using the CFC editor. You also need to specify the task to be used to compute the individual function blocks.

Parameterization dialogs in the CFC editor

You can double-click on the function block header, or select **Edit > Object properties**, in order to override the following configuration defaults:

Table 2- 5 Configuring function blocks

Specifications	Description
General	You may configure the name and comment to be displayed in the function block header. In the "Special object properties" area, you can perform all steps that are necessary to prepare a block for operation and monitoring in WinCC.
Runtime properties	In this area, you can modify the task that contains the runtime sequence of a function block, which was specified in "Insert function block". You may "search" for the selected function block in the runtime sequence and "remove" and "reinstall" it at a different location.
I/Os	In this area, you can enter following I/O data for all parameters: <ul style="list-style-type: none"> • Values and comments for I/O parameters • Visibility in the CFC of parameters that are not interconnected • Setting the ID of parameters to be enabled or disabled for testing • Scaling value for parameters of data type REAL • Unit texts

You can find **more information** on the topic of "Interconnecting function blocks" in the D7-SYS - STEP 7, configuring CFCs and SFCs (<http://support.automation.siemens.com/WW/view/de/8776786/0/en>) manual.

Specifying runtime properties

Several function blocks calculated in sequence in a task can be combined to form a runtime group. Along with the task structuring option, this solution allows you to individually enable/disable processing.

Note

When a runtime group is disabled by setting its interconnected function block input accordingly, all function blocks contained therein are no longer computed.

The engineer specifies the runtime properties of the function blocks by assigning these to a cyclic task, or to an interrupt-controlled task, or to a runtime group and by defining their position in the task or runtime group. These properties are decisive for the response of the target system with regard to the following factors:

- Deadtimes
- Response times
- Stability of time-dependent structures.

Assigning the function blocks to cyclic tasks

You can assign the function blocks to one of the five cyclic tasks by calling them in the CFC editor, or by editing the program code of the runtime sequence in the CFC editor. Each function block can therefore be assigned to a cyclic task and processing sequence within the sampling time of the task.

Assigning function blocks to interrupt tasks

In order to enable processing of function blocks with interrupt control, the blocks are entered in one of the eight possible interrupt tasks in the user-specific at the time of their call, or in the program code of the "Runtime sequence of the CFC Editor". Computing of individual function blocks can be initiated by means of a specific hardware interrupt.

Note

In contrast to cyclic tasks, the interrupt tasks are not started at constant bus cycle times and are rather initiated by hardware interrupt events.

CPU555 can also be used to synchronize interrupt tasks on the isochronous PN line.

In addition, interrupts can be called at a constant bus cycle time. You can use the PAC function block (Process Alarm Counter) to trigger a processor interrupt when the run time limit is reached. The function block PAC (Process Alarm Counter) assigns parameters for counters C1 and C2.

Configuring equivalent sampling times

Certain function blocks, e.g. control blocks, must to be processed at cyclic intervals due to the program design. If these are to be configured in an interrupt task, an equivalent sampling time must be configured in the HW Config program section for this particular interrupt task. This should approximately correspond to the average interval between two interrupt task calls.

To configure the equivalent sampling time, double-click the module and select the "Interrupt task" menu command.

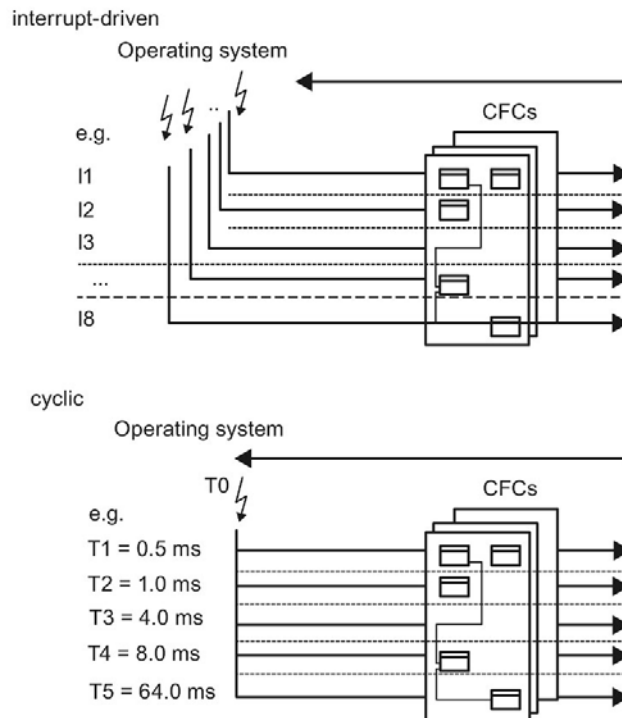


Figure 2-1 Processing of the function block by the operating system of CPU551

Processing the function blocks

In SIMATIC TDC, the control task can be implemented by interconnecting and assigning parameters to the function blocks with orientation on the block diagram. A function block type can be used at multiple instances. The function blocks are parameterized and interconnected at the block I/O.

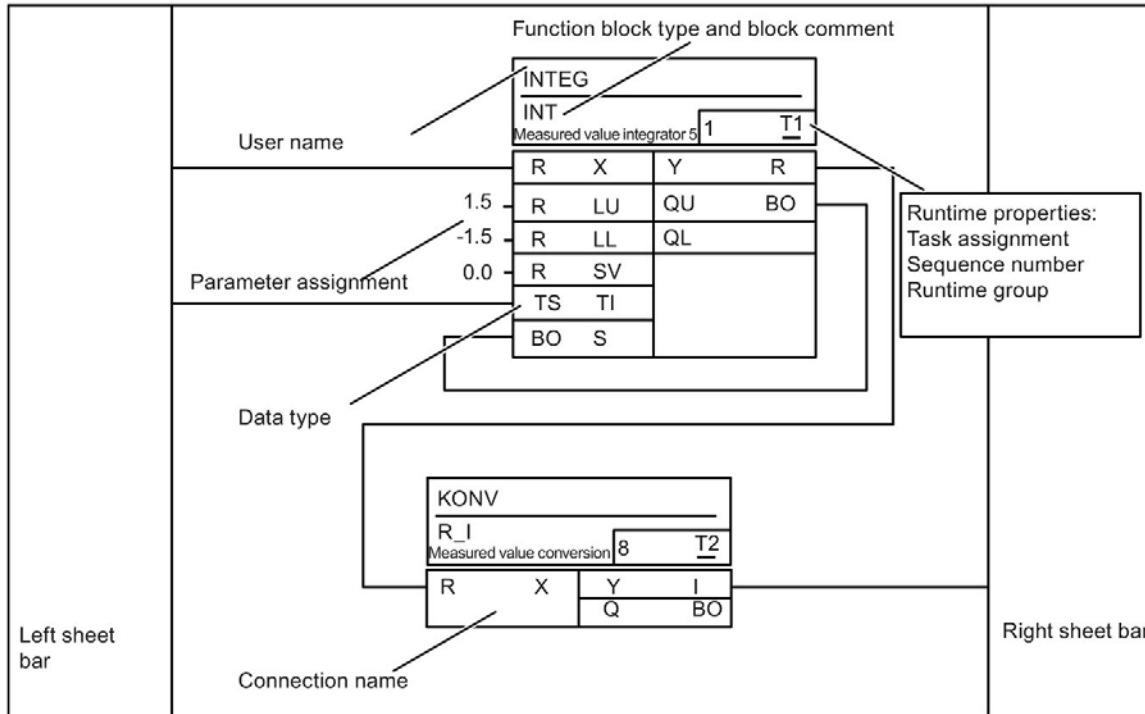


Figure 2-2 CFC sheet - work area

The following I/O is available for configuring and interconnecting the function blocks:

- Input connections (function block inputs)
- Output connections (function block outputs).

Input connections

The project engineer can parameterize the inputs using constants, or interconnect these with other function block outputs. The I/Os of the function blocks called are already assigned a default configuration that you can change.

Output connections

You can interconnect the outputs with other inputs, or assigned these an initialization value that differs from the default value. This value is available at this I/O at the time the function block is computed initially in the INIT operating status. This is useful if you want to explicitly preset the output of a flip-flop block.

Margins

The margins on the left and right of a CFC sheet contain cross-references to the interconnected objects, e.g. other blocks, or runtime groups that are not included in the current sheet. Moreover, they also contain the number of the connector (termination location), if the Autorouter is unable to draw the connecting line to the margin because the sheet is overfilled.

Overflow sheets

Overflow sheets are automatically created if the number of sheet bar entries generated on a sheet exceeds the space available on the sheet. An overflow sheet consists only of the sheet bars and contains no other objects.

Parameterizing

Instead of an interconnection, you may parameterize a constant for each input or output a constant in order to override the defaults.

A block I/O can be identified as parameter by means of pseudo comment.

Interconnecting

The term interconnecting denotes the following:

- Connecting a function block output to another function block input on the same CPU.
- Interconnections of a function block output to a runtime group
- Interconnection of a function block output to a global operand, or to a global operand with function block input. A global operand may represent the following:
 - A name with a "\$" dollar prefix, i.e. interconnection of a signal to or from a function block of another CPU.
 - A virtual connection name or virtual connection, i.e. a transfer of process data between function blocks via any coupling modules using the process data service.
 - A symbolic hardware address. In this case, a hardware address represents the symbolic name of one or several associated terminals of a module, e.g. binary inputs of an I/O module. The symbolic hardware address is defined in the HW Config program section.
 - A name reference, i.e. the name of a message system.

All types of interconnections that exit a chart sheet generate an appropriate cross-reference in the margins of the CFC sheet.

Comments

Each function block I/O on the CFC can be provided with a comment.

Pseudo comment

There is a pseudo comment which is identified by the @ character as prefix and can be separated by space characters from the following standard comment text:

@DATX

The input is interconnected while bypassing the consistency mechanisms (see the section "Description and usage of signal transfer mechanisms (Page 44)").

2.1.2.3 The third step: Compiling the user program and loading it into the CPU

Once you have completed configuration of all necessary hardware modules in HW Config and set up the relevant function blocks in the various charts using the CFC editor, you can use the compiler to translate the software into the CPU's machine code. Once this has been completed, you have two choices:

Offline download

A memory module (CPU550/551 MC / CPU555 MMC) is programmed using the PCMCIA interface of the configuration PC. The modules are ready for operation as soon as all of the correctly programmed memory modules of all CPUs in the rack have been inserted.

Online download

User program and operating system are loaded directly from the configuration PC to the CPU as follows:

- CPU555
 - Via local PROFINET IO interface
 - Via another CPU555
- CPU551:
 - Via the serial interface (DUST1)
- CPU551/555:
 - Via CP51M1
 - Via CP50M1 (MPI)

Initial loading to a CPU

To load a user program into a CPU, the IP address of online access (CPU555 or CP50M1) must be set correctly before loading.

If the current IP address of a CPU555 or CP50M1 does not match the configured IP address in HW Config, loading into a CPU555 or CP50M1 is not possible. The error message "Online: A connection could not be established" then appears. This applies in the following cases:

- A brand new CPU555 or CP50M1 does not yet have an IP address.
- A CPU555 or CP50M1 was already used in another project and has a different IP address.

Setting the IP address

Follow these steps to set the IP address of a CPU555 or CP50M1:

1. Set the module to user stop state "d" or initialization error "0".
2. Connect the module online.
3. In the SIMATIC Manager, open the dialog under Target system > Edit Ethernet Node...
4. Click "Browse" under "Nodes accessible online".
5. Select the appropriate MAC address.
6. Enter the appropriate IP address and subnet mask.
7. Click "Assign IP configuration".
CFC projects can now be loaded after successful transfer.

2.1.3 Operating states of a CPU module

The SIMATIC TDC system may enter the states listed in the following table:

Table 2- 6 System states of a CPU module

Operating state	Power off	INIT	RUN	STOP		
				User stop	Initialization error	System error
Internal system status						
Status description	De-energized state	System startup (initialization)	Cyclic operation (normal operation)	Stop mode triggered by user	Status after init error	Status after fatal system error
Properties	System not in operation	System startup --> no external influence possible	Functionality in accordance with the configuration	No cyclic processing -> fast download	Incorrect initialization --> no transition to cyclic operation	Fatal system error -> processing aborted
Seven-segment display	Off	'0'	CPU number ('1' ... '20'; flash test for identification) or 'C', 'E', 'b', 'A'	'd' (Permanent "d": Module is in STOP, triggered by another CPU. Flashing "d": Module is in STOP, triggered by this CPU itself).	'0' (Flashing "0": Error on this module. Permanent "0": Error on a different module)	'H' (Flashing "H": Error on this module. Permanent "H": Error on a different module)
Available diagnostic interfaces	--	None	All configured (one must be on the first CP) and local interfaces		Local interface and first CP interface	Local interface
Available control functions	--	None	Full functionality of CFC online	Diagnostics or download only	Diagnostics or download only	Diagnostics or download only
Management via user interface (CFC)	--	--	States can be queried via a dialog.			

Definition of terms

Term	Description
First CP interface	SIMATIC TDC: Interface that is inserted at the top in the first CP50M1 / CP51M1 in the rack (counted from left to right).
Diagnostics	Only possible to read error fields

INIT, RUN, and STOP represent the operating states, whereby STOP is split into three different system states.

User stop system state (not with CPU550)

The "User stop" system state is used to quickly load a program via CP50M1/CP51M1 or the local interface of the CPU. Quick in this context means that cyclic processing is stopped in this state and all CPU resources are made available for the download. In the seven-segment display, a "d" blinks when a user stop is requested for the CPU. This state is initiated by the user; the assigned parameters of the configured diagnostics interfaces (CP50M1/CP51M1) remain valid. A user stop results in a stop of all CPUs in the same rack.

Download in RUN

It is also possible to perform a download in RUN using any utility, but with significantly extended download times (data is downloaded while cyclic processing is also active).

The transition from RUN to "User stop" status is only possible if a user explicitly requests this status via (local, or configured) service interface. In this state, all configured service interfaces and the local service interface remain available, which means that diagnostics and downloads are still possible via all service interfaces (this is necessary if several PCs are connected to the rack).

2.1.4 Description and usage of signal transfer mechanisms

The term signal transfer denotes data exchange between different blocks.

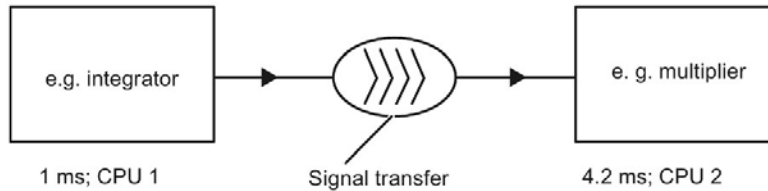


Figure 2-3 Data exchange between two tasks

2.1.4.1 Data consistency

For interconnections between different cyclic tasks, SIMATIC TDC ensures consistency of all data which is transferred. This means that all data transferred from a task originate from the same computing cycle of this task. All values calculated during a scan cycle are "exported" at the end of the task. The necessary values are "imported" at the start of a task, whereby it is ensured that there is no time-related overlap between read and write access to the values (buffer system). As deadtimes are unavoidable with this concept, a signal should not be routed unnecessarily via several tasks and CPUs.

A differentiation is made between the following signal transfer types:

- Data exchange within the same task of a CPU
- Data exchange between different tasks of a CPU
- Data exchange between cyclic tasks of several CPUs
- Data exchange between interrupt tasks of several CPUs

2.1.4.2 Data exchange within the same task of a CPU

Each function block output in the system is assigned a memory cell that the function block uses to save its currently computed value during block processing. All inputs that are interconnected with the outputs in the same task fetch their values from the corresponding memory cells assigned to the respective interconnected output. In order to prevent deadtimes, the blocks of a task should be computed according to the "signal flow", i.e. the first block to be computed contains the outputs that serve as inputs for a successor block, etc.

2.1.4.3 Data exchange between different tasks of a CPU

Data exchange between different tasks of a CPU is handled by means of a buffer system in order to ensure data consistency (refer to "Data consistency (Page 44)"). However, whenever data is exchanged between a faster and slower task, it should be noted that value changes may be registered late or not at all in the slower task. If this cannot be tolerated, the configuration must be adapted, e.g. using pulse stretching function blocks.

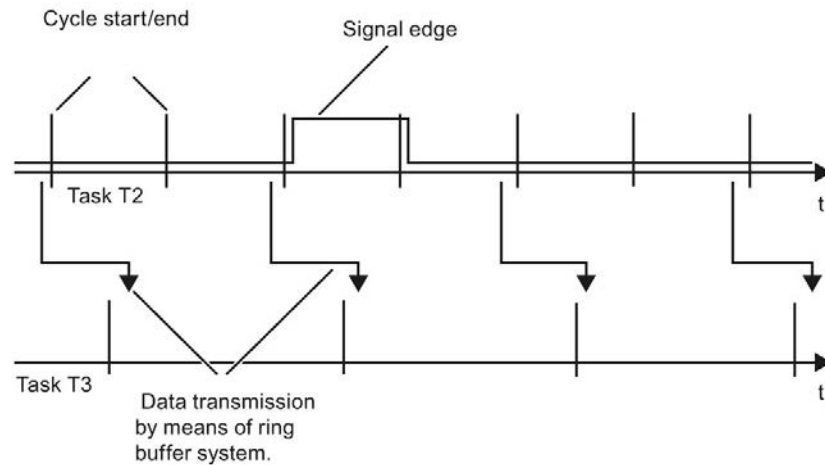


Figure 2-4 Signal not registered in task 3

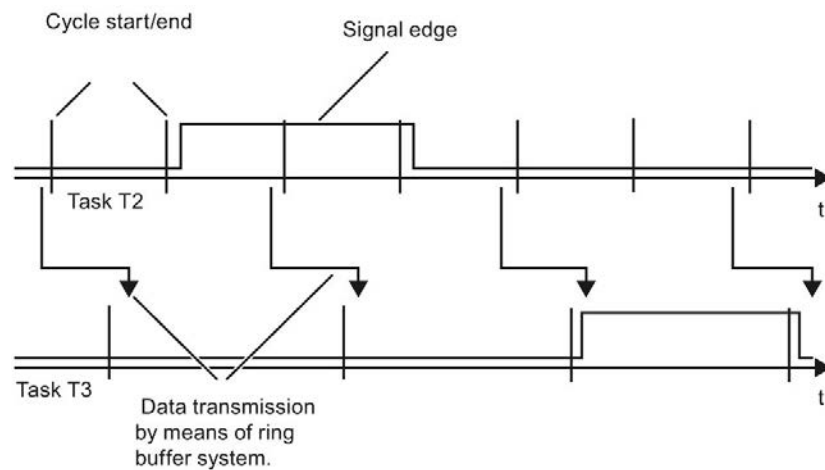


Figure 2-5 Late signal registration

2.1.4.4 Data exchange between cyclic tasks of several CPUs

The signals are transported between the CPUs by means of the communications modules CP50M1/CP51M1 (SIMATIC TDC). \$ signals are used to handle the connections between function blocks that run on different CPUs in the same station (menu command "Insert-connection to the operand" in the CFC Editor). You need the following data to configure a \$ signal:

- Signal name
- Type
- Bus assignment.

The dollar signal type defines whether data transfer is to be

- consistent ("standard") or
- inconsistent ("fast \$ signal")

For a fast \$ signal, you can always access a current value. The dead time generated during signal transfer is reduced to a minimum if the generator (source) and consumer (destination) are configured in the same task and the tasks are synchronized (see section "Significance and application of CPU synchronization (Page 54)").

Note

If time-critical functions are processed on the CPUs of a rack, you should observe the following rules:

- If you use \$ signals, you must ensure that every \$ signal is used.
 - Limit the number of \$ signals to a minimum.
 - If possible, configure all communication links of the rack coupling on one or a maximum of two CPUs of the rack.
 - Configure the CPUs containing the configured communication links of the rack coupling so that there are no additional CPUs between these CPUs and the rack coupling module, if possible.
-

2.1.4.5 Data exchange between interrupt tasks of several CPUs

Fast \$ signal

A fast \$ signal must always be configured if the signal is generated or used in an interrupt task, since an interrupt event can occur only once at any point in time, which makes it necessary to bypass the consistency mechanisms in order to prevent data loss. A conflict could occur between the request for data consistency and low dead time, which must be decided according to the application.

Note

It should always be checked as to whether problems could develop if there is no data consistency.

Data consistency can be achieved by "looping the signals" through a cyclic task on the CPU module that is used to calculate the interrupt task. The deadtime computation is illustrated in the following table.

Table 2- 7 Calculation of deadtime values

Time interval	Calculation
Minimum value	$1 \times T_x$
Maximum value	$2 \times T_x + 1 \times T_y + 1 \times T_{\text{alarm}}$

- T_x = sampling time of the cyclic tasks through which the signals are looped
- T_y = sampling time of the source/destination CPU
- T_{alarm} = maximum interrupt repeat time of the interrupt task.

2.1.4.6 Minimizing deadtimes

To minimize deadtimes, a signal can be directly transferred while bypassing the data consistency mechanism. In this case, it is "wired" directly to the output of the generating block. You have two options of configuring the deadtimes:

- Pseudo comment @DATX for interconnecting tasks of a CPU
- Fast \$ signals for interconnecting several CPUs

2.1.4.7 Processing sequence within a basic CPU clock cycle

The Task Manager (see "Illustration of the mode of operation of the Task Manager (Page 60)") of the operating system is started with the basic CPU clock cycle T0. This determines the tasks to be started (T1 and maximum of one other Tn, with Tn from {T2...T5}).

The following components may have to be executed within task processing:

- Buffer changeover for the tasks to be started (T1 and, if necessary, an additional task Tn)
- System mode of the blocks in T1 corresponding to the block sequence (see section "Significance and application options of the process image (Page 50)")
- System mode of the blocks in Tn corresponding to the block sequence (see section "Significance and application options of the process image (Page 50)")
- **Importing signal interconnections in T1 and standard mode T1**
- **Exporting signal interconnections from T1**
- **Importing signal interconnections in Tn and the standard mode Tn**
- **Exporting signal interconnections from Tn**

The components relevant for signal transfer are highlighted in bold.

Note

Synchronization to basic clock cycle

When synchronizing the CPU555/CPU551 to a basic clock cycle received from a CP53M0, please make sure that the sampling time set for the CPU555/CPU551 corresponds to this basic clock cycle. Otherwise, there could be problems on the received basic clock cycle during synchronization, e.g.:

- Complete failure of synchronization
 - Incorrect calculation of the time-dependent blocks such as BF, PT1, or DIF
 - Changing cycle times for the CPU555 during sampling times
 - Occurrence of cycle errors
-

2.1.4.8 Interconnection changes and limited number of interconnections

Interconnection changes in the configuration test phase

Interconnections extending beyond the task boundaries can only be changed with certain restrictions using the test mode of the CFC editor. The CFC editor test mode is used to test and optimize the user program that is already executed online on the CPU.

When the service makes such changes, there are only a limited number of reserves for additional interconnections. The number of additional interconnections available for reserve amounts to

- 20 % of the configured number of interconnections,
- but at least 10.

Example:

There are already 5 interconnections from cyclic task T2 to cyclic task T3. For interconnection changes from T2 to T3 there is then a reserve of 10 interconnection changes, The 20 % of 5 = 1 additional interconnection as in rule 1 would otherwise fall short of the minimum reserve of 10 additional interconnection as in rule 2.

For 100 existing interconnections, there are an additional 20 reserve interconnections, as 20 % of 100 = 20.

Limited number of interconnections

A differentiation is made between interconnections within a task, between tasks of a CPU, and between several CPUs of a station. For operation with several CPUs, an additional differentiation is made between standard and fast \$ signals.

For interconnections between tasks of a CPU, the alternating buffer system on the processor is used. The maximum number of interconnections is limited by the main memory expansion stage.

Connections between several CPUs of a station are handled via the memory buffer modules. The number of possible interconnections is dependent on the memory buffer module and signal types used.

You can find **more information** on coupling memory modules in the manual "SIMATIC TDC hardware (<http://support.automation.siemens.com/WW/view/de/8776697/0/en>)".

Note

At least 500 \$ signals can be configured. The maximum number depends on the configuration.

2.1.5 Significance and application options of the process image

A process image represents a snapshot of all interface signals from the process at the start of a cyclic task.

Necessity for data consistency

On a digital control system, the interface signals must be processed consistently to the individual processes. In this case, interface signals are the digital and analog input or output signals of a hardware module.

The input signals of the different tasks must be kept constant during a computation cycle. If this was not the case, interface signal changes while processing a task and runtimes of the individual function blocks would unpredictably influence the result of a computation cycle.

The data from the hardware interfaces is processed in the "process image", implemented by the system mode of the function blocks when a task is started to be processed.

The Task Manager (refer to chapter "Mode of operation of the Task manager") of the operating system is started with the basic CPU clock cycle T_0 . This determines the tasks to be started (T_1 and a maximum of one additional T_n , with T_n from $\{T_2...T_5\}$).

Task processing

The following components may have to be executed within task processing:

- Buffer changeover for the tasks to be started (task 1 T_1 and, if necessary, an additional task T_n)
- **System mode of function blocks in T_1 corresponding to the block sequence**
- **System mode of function blocks in T_n corresponding to the block sequence**
- Importing signal interconnections in T_1 and standard mode T_1
- Exporting signal interconnections from T_1
- Importing signal interconnections in T_n and the standard mode T_n
- Exporting signal interconnections from T_n

The components relevant for the process image are highlighted in bold; for the other components, refer to chapter "Description and usage of signal transfer mechanisms".

2.1.5.1 Implementing the process image

System mode

The system mode is used to realize the process image before a task is computed. The following figure illustrates in which cyclic mode sequence (CPU in RUN state) the function blocks are computed in system and standard mode. In this example, functions blocks 10 and 30 are computed in system mode within the framework of the process image so that the results can be subsequently consistently used in the standard mode.

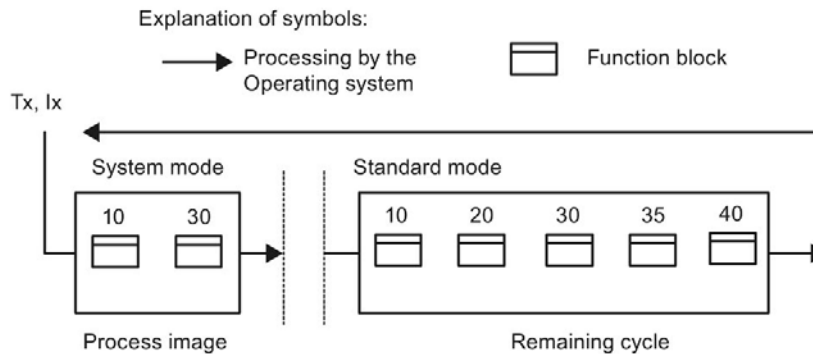


Figure 2-6 Sequence of the function block calculation in the system and standard modes

The system mode starts immediately after the initiating event (hardware interrupt or basic clock cycle) in order to generate a real-time process image. The execution between the jump into the operating system up to the end of the system mode can only be interrupted by a higher priority system mode. Among other things, function blocks with access to the I/O devices are computed.

2.1.5.2 Process image for cyclic tasks

Input blocks with system component

For input blocks, which have a system component or whose system component is activated, the input signals are read-in from the hardware and buffered. The signals are evaluated in the standard mode of blocks of the same cycle.

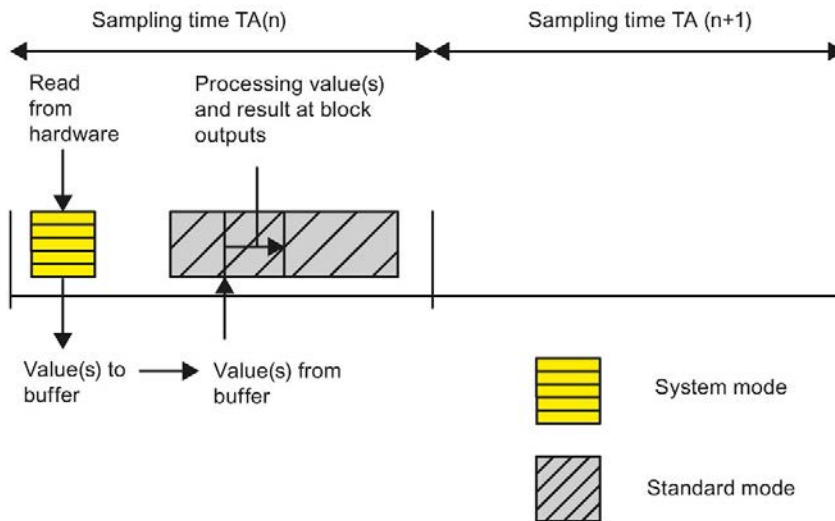


Figure 2-7 Sequence of the system mode for input blocks

Output blocks with system component

For output blocks that have a system component or whose system component is activated, the signals to be output are calculated in the standard mode of the previous cycle corresponding to the block function and the actual I/O values. These signals are buffered. Output to the hardware is in system mode at the start of the next sampling cycle.

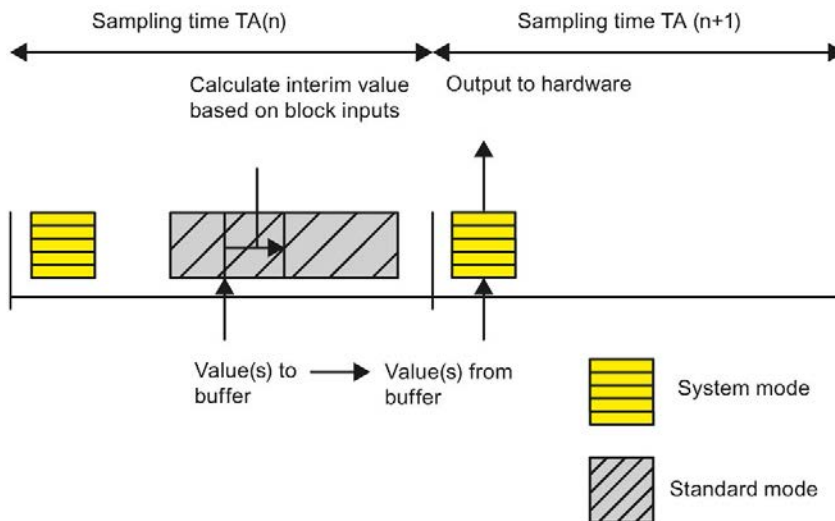


Figure 2-8 Sequence of the system mode for output blocks

As the system component is essentially restricted to the input and output of hardware signals, the system mode is processed within just a few micro seconds.

For several input/output blocks, the „DM“ block input can be used to control whether an input/output is made in the system mode or in the standard mode. With calculation in **standard mode**, the interface signals at the blocks are computed in **standard mode** while bypassing the process image. For input blocks, the signals are read-in immediately before being computed, and for output blocks, immediately after their computation.

2.1.5.3 Process image for interrupt tasks

An interrupt task has essentially the same behavior as a cyclic task.

Mode of operation of an interrupt task

A higher-priority task can interrupt an ongoing lower-priority task (see section "Basic functions of the operating system (Page 65)"). This means that, e.g. for longer computation times of an interrupt task, the start of cyclic tasks and therefore output to the hardware may be delayed, as the signal is only output to the hardware after the next task has been started for output blocks with system mode.

Moreover, it should be precisely checked when using input/output blocks with the system mode within an interrupt task for non quasi-cyclic interrupts. In this case, the output is only realized after the next interrupt event whose timing is unknown. For specific input/output blocks, this problem can be remedied by using a block input so that input/output is realized in the standard mode.

2.1.6 Significance and application of CPU synchronization

Configuring CPU synchronization

CPU synchronization is configured in the HW Config program section. The directory of the corresponding SIMATIC TDC station is opened in SIMATIC Manager and HW Config is activated by double-clicking on the hardware icon in the right section of the window. Now select the required CPU module. There are separate dialog windows to synchronize the basic sampling time of the CPUs and the interrupt tasks under **Edit > Object characteristics**.

Synchronization mechanisms

SIMATIC TDC provides the following synchronizing mechanisms:

- Time synchronization
- Synchronization of the own basic clock cycle to the basic clock cycle of a master CPU
- Synchronizing the own basic clock cycle to an interrupt task of a master CPU
- Synchronizing the own interrupt tasks to interrupt tasks of a master CPU
- Synchronizing several stations
- Synchronization on PROFINET clock cycle
- Response to synchronization failure
- Configuring synchronization of the CPU basic clock cycle
- Configuring interrupt task synchronization

2.1.6.1 Time synchronization

The real-time clocks of all CPUs in a SIMATIC TDC station are synchronized to the clock of CPU inserted at slot 1 to prevent the various CPU clocks from drifting apart. This synchronization is automatically executed at cyclic intervals of 10 s.

Note

Time synchronization via a SNTP signal

A CP51M1 is required for time synchronization via a SNTP signal.

The following conditions must be met to use the SNTPR block:

- The block expects a time server to send an (S)NTP frame to UDP port 123 of the CP51M1 every 10 seconds. SNTPR does not send any queries to a server.
- The (S)NTP frame must be sent to the CP51M1 by broadcast (not multicast).
- The (S)NTP frame is only accepted with accuracy stratum 1 or 2.
- The (S)NTP frame should arrive in a 10-second cycle with the greatest possible accuracy because different configuration is not possible.
- Deviations from this cycle time have a direct effect on the accuracy of the TDC time of day.

2.1.6.2 Synchronization of the own basic clock cycle to the basic clock cycle of a master CPU

The basic clock cycle can be switched from a CPU to the L and/or C bus of the rack and can be received from other CPUs of the station, or by several SIMATIC TDC stations, which are coupled using the rack coupling or GDM coupling. For the receiving CPU, an offset can be configured between the basic sampling time and the sender basic sampling time. This time offset can also be changed online while the CPU in the RUN state using the DTS function block type.

2.1.6.3 Synchronizing the own basic clock cycle to an interrupt task of a master CPU

It is possible to trigger a bus interrupt at the start or at the end of an interrupt task of a transmitting CPU. This can be received from one or several other receiving CPUs where it is then used to generate the basic clock cycle.

2.1.6.4 Synchronizing the own interrupt tasks to interrupt tasks of a master CPU

To synchronize an interrupt task, it is possible to use a bus interrupt, initiated at the start or the end of an interrupt task from a transmitting CPU. This interrupt can be received at one or several other receiver CPUs in order to initiate an interrupt-controlled task there.

2.1.6.5 Synchronizing multiple SIMATIC TDC stations

The modules CP52M0, CP52IO, CP52A0 (SIMATIC TDC) and CP53M0 (SIMATIC TDC with SIMADYN D) are available for cross-station synchronization of the basic sampling time. In this case, the bus systems of the two stations are connected via coupling modules.

2.1.6.6 Response to synchronization failure

The basic clock cycle is monitored on the synchronized receiver CPUs using a hardware timer. If the transmitted clock is no longer available for 4 cycles, the basic clock timer on the CPU module generates the basic clock cycle. The basic sampling time configured in HWConfig is used as basis, which in this case serves as the equivalent sampling time. The changeover to the basic clock cycle of the CPU is signaled by a flashing "E" on the 7-segment display of the CPU module, and is flagged in the error field. When the external clock source kicks in again, this can be again used on the basic sampling time clock receiver using the "DTS" function block type.

2.1.6.7 Configuring synchronization of the CPU basic clock cycle

The configuration is set in the "Basic clock cycle" dialog window of HW Config (refer to the section "Significance and applications of CPU synchronization"). Synchronization is disabled by default.

Basic clock cycle generated by the CPU itself

If the CPU should generate a basic clock cycle itself, the following settings must be completed in the "Basic clock cycle" dialog:

- Activate the "Generate" button with mouse click.
- Enter the required basic sampling time from 0.1 to 16 ms.

In the lower section of the window, it can be defined as to whether the selected CPU should be used as the source for the basic clock cycle. The appropriate bus must be set for this purpose. „No“ is set by default.

Synchronizing the basic clock cycle to a source.

If the basic clock cycle is to be synchronized to another source, HW Config requires the following settings:

- Activate the "Synchronize" button with mouse click.
- Select the required source from a list, e. g.
 - Bus interrupt (SIMATIC TDC)
- Enter an equivalent sampling time of 0.1 to 16 ms.
 - Default = 1.0 ms
- If necessary, enter a synchronization delay time of 0.1 up to the equivalent sampling time.
 - "None" is set by default for the sampling time

2.1.6.8 Configuring interrupt task synchronization

This configuration is made in the "Interrupt tasks" dialog window of HW Config. Synchronization is disabled by default, i.e. no hardware interrupts are defined and no bus interrupt is sent.

Setting interrupt task synchronization

- The mouse is used to set one of the eight possible interrupt tasks I1 - I8.
- Select the required source of the defined hardware interrupts from a list, e.g.
 - Bus interrupt
 - CPU counter C1 or C2
 - PROFINET IO clock cycle
- Enter an **equivalent sampling time** of 0.1 to 16 ms.

CPU as interrupt source for the rack

The lower window section specifies whether or not the selected CPU is to function as hardware interrupt source for the rack. In this case, select one of the defined interrupt tasks I1 - I8 for transmission. You can also select whether to transmit the interrupt task at the start or end of interrupt task processing.

Transmitting at the start of interrupt task processing

It is practical to transmit the interrupt task at the start if an alarm interrupt is to start synchronously and without delay on several CPU modules. However, the interrupt task on the receiver CPU module may end before the interrupt task on the transmitting CPU module because the transmitting task has been blocked by a higher-priority interrupt.

Transmitting at the end of interrupt task processing

Transmitting at the end ensures that the task on the receive side is never started before the transmit task is completed. It is practical to use this second option for data transfers from a transmit to a receive task.

2.1.7 Significance of the processor utilization

2.1.7.1 Determining the approximate processor utilization

The CFC calculates a value for CPU computing time utilization during compilation. For this purpose, it accesses a list that contains the computing time slice for each function block type. These computing times were calculated for the "worst case" scenario in the block development phase and are available in the programming manual "SIMATIC D7-SYS Selecting function blocks (<http://support.automation.siemens.com/WW/view/de/14952400/0/en>)" and in the online help.

A typical computing time slice is used (e.g. for medium bus load) as the worst-case scenario would lead to intolerable time slices particularly under the aspect of function blocks that access hardware. Given these guideline values, the actual computing time of certain function block types may fluctuate considerably.

The computing time entered in the "SIMATIC D7-SYS Selecting function blocks (<http://support.automation.siemens.com/WW/view/de/14952400/0/en>)" programming manual specifies the typical computing time of the block in μs . However, particularly at communication blocks this value may deviate considerably from the actual time required to transmit the specific data volume.

Once you have compiled CPU charts in the CFC editor by means of **Chart > Compile** menu command, the path of a MAP listing is specified in an info window or error window.

For the reasons mentioned above, the processor load value entered in the MAP listing represents a guide value that usually has a tolerance of approx. +/- 10 %.

2.1.7.2 Precise calculation of processor load

Function block PSL

CPU load can only be calculated based on the configuration of the "Permanent System Load" PSL block. The PSL block is configured in any cyclic task in the CPU to be analyzed.

The PSL block has 5 outputs (Y1..5) that display the current load of the various tasks in the form of a load factor. The displayed factor should not exceed 1.0 (100%). Values exceeding 1.0 indicate that CPU overload.

In addition, the PSL block has 5 inputs (T1..5) that can be used for each task to simulate additional loads in milliseconds (ms). You can then take a reading from these outputs to determine the effect of such loads of individual tasks. Load is calculated by measuring the runtime of the tasks and then dividing the result by the actual sampling time. Task runtime includes higher-priority tasks that extend runtime and seemingly increase load. It is therefore impossible to obtain a total load value by adding up these values.

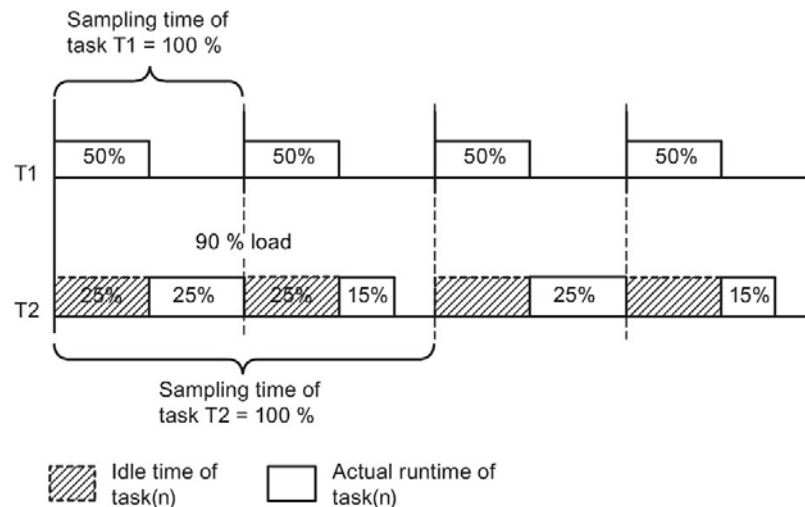


Figure 2-9 Calculating runtime

2.1.7.3 Illustration of the mode of operation of the Task Manager

The figure in this chapter illustrates the mode of operation of the Task Manager.

The illustration of the first cycle indicates that all tasks can be completed within a basic sampling time thanks to low computing time load.

A task that cannot be completed within a basic sampling time due to increased computing time is completed within the next basic cycle clocks. The tasks with short sampling times are completed before tasks with long sampling times, i.e. T1 before T2, before T3 before T4 before, etc. This distribution is allowed, i. e. without cycle error, as long as the required sampling times are maintained (see the second and third cycle).

Cycle errors

If computing time load rises to a still higher level, a cycle error will eventually occur at the task with the longest sampling time. This means that the sum of the function blocks cannot be computed completely within the configured sampling time.

Note

An "E" error ID is set if a specific number of cycle errors is exceeded and is displayed on the 7-segment front panel display of the CPU, if this is the highest priority error status of the CPU at this time.

In addition to the configurable interrupt tasks, the cyclic tasks are interrupted particularly if communication goes down. These interrupts ensure, for example, that the data to be transmitted and received via the serial interfaces is processed before new data is received. Transmit and receive interrupts such as these may occur independently of the configured cycle time of the appropriate communication blocks at almost any instant in time. This situation, in combination with random occurrence of interrupt tasks and exceptionally high processor load, can basically lead to a task overflow triggering one or several cycle errors at cyclic tasks.

Particular attention should be paid to this aspect if

- load generated by the task with the shortest sampling time is extremely high and
- the functions computed in this task are extremely sensitive to sporadic sampling cycle failure (e.g. position controls).

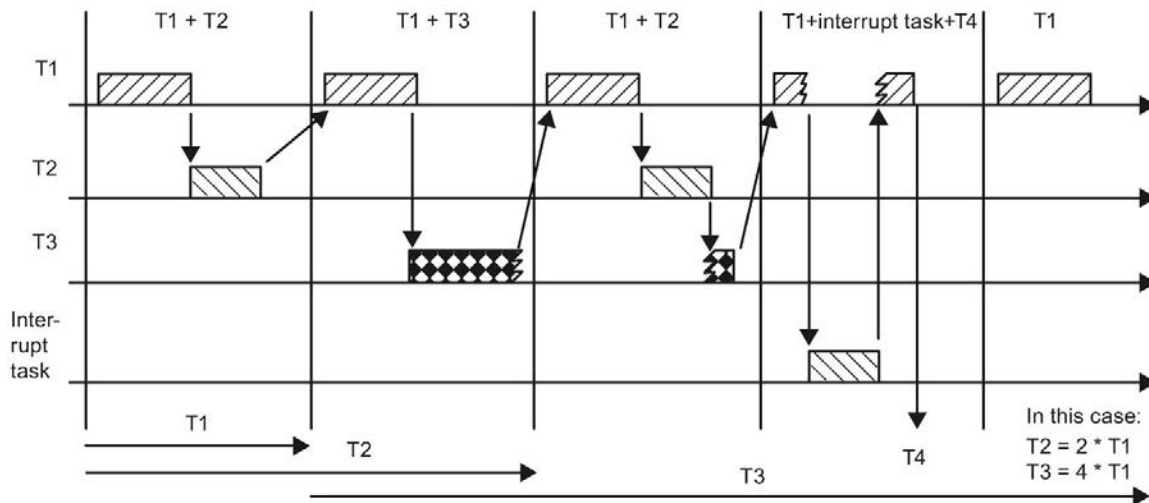


Figure 2-10 Sequence of a configured task

2.1.7.4 Eliminating cycle errors

The modular SIMATIC TDC provides the following options of eliminating cycle errors:

- Extending the configured basic sampling time
- Moving configured blocks from fast to slow tasks.
- Using several or higher-performance CPUs or multiple SIMATIC TDC stations
- Reducing the number of blocks, or changing the block types
- Checking the necessity of communication interfaces on this CPU
- Checking the necessity for interrupt function packages on this CPU

Note

Determine the most cost-effective way to achieve the desired result on a case-to-case basis.

2.1.7.5 Error displays on task overflows

The following table shows the error response in the event of overflows of interrupt tasks and cyclic tasks:

Table 2- 8 Incorrect response to overflowing tasks

Type of overflow- ing task	Situation	Display on CPU		
		No error display	"E"	"H"
Cyclic	User program generates individual/one-time overflow	X		
	User program generates 3 overflows at once		X	
	User program generates 100 overflows at once			X
Interrupt	User program generates individual/one-time overflow		X	
	User program generates 3 overflows at once		X	
	User program generates 100 overflows at once			X

The handling of overflows differs in the case of the message "E". With cyclic tasks, an "E" is only reported after three overflows in succession. With interrupt tasks, the first overflow results in an "E". The reason for this is that the interrupt could be a one-time event that may not reoccur immediately. If the associated task is not executed, this is indicated.

In the event of an overflow, regardless of the type of task, the start command for the task is discarded and not resent.

Note

Using a small user program which runs in the T5, you have the option of acknowledging the display of "E" from an application and, for example, reducing the sensitivity with a dedicated counter.

2.1.8 Technical specifications of the operating system

2.1.8.1 Performance features

The most important properties and technical specifications of the operating system are specified in the following.

You can find detailed specifications of the CPUs in the system manual "SIMATIC TDC hardware (<http://support.automation.siemens.com/WW/view/de/8776697/0/en>)".

Number of CPU modules

A CPU module requires one slot. Slots not in use by CPU modules can be used for peripheral modules.

A maximum of 20 CPU551 can be plugged into one rack. When GDM coupling is used, a maximum of 19 CPU551 can be inserted in one rack because a CP52A0 is needed for rack coupling.

The CPU module CPU555 can only be operated in the rack UR6021. A maximum of eight CPU555 can be inserted in one UR6021.

Number of function diagrams

The maximum number of function charts depends on the particular software, but is approximately 65536 (2^{16}).

Cyclic tasks

System chart	Available automatically
Basic sampling time T0 can be configured	CPU551: From 0.1 to 16 ms in steps of 0.1 ms CPU555: From 0.1 to 16 ms in increments of 0.005 ms
Number of configurable cyclic tasks	5
From	CPU551: T0 CPU555: 1/4 x T0; 1/3 x T0; 1/2 x T0; 1 x T0
To	CPU551: T0 x 2 ¹⁵ CPU555: T0 x 2 ¹⁶
Configurable	CPU551: T0 to 32768 x T0 CPU555: 1/4 x T0; 1/3 x T0; 1/2 x T0; 1 x T0 to 65535 x T0

Alarm tasks (CPU551)

Number of configurable alarm tasks		8
Total number of available alarm sources		16 (SIMATIC TDC)
Of which		
	Software interrupts	8
	CPU timer interrupts	2
	Interrupts for binary inputs	4
	Bus interrupt	1 (SIMATIC TDC)
	Extended LE bus interrupt X1	1

Alarm tasks (CPU555)

Number of configurable alarm tasks		8
Total number of available alarm sources		22 (SIMATIC TDC)
Of which		
	Software interrupts	8
	CPU timer interrupts	2
	Bus interrupts	1
	PROFINET alarms	9 (SIMATIC TDC)
	Extended LE bus interrupt X1	1
	SB591_IRQ	1

Computing times of the operating system

The following table shows the minimum time for processing task cycles (see above for calculation basics!):

Time to start	40 µs
Time to end	40 µs
Additional component with local buffer system	20 µs

2.1.8.2 Basic functions of the operating system

Operating system components

The operating system consists of the following components:

- Task Manager for cyclic and alarm-controlled processing
- Hardware and software initialization
- Memory management (buffer management)
- Operating system data and OS lists
- Coupling to the other components (system interfaces)

The operating system is capable of multi-processing and multi-tasking.

The basic operating system functions are embedded in the overall system and represent the most important interfaces to the environment.

Table 2- 9 Basic functions of the operating system

Operating system functions	Triggered by	Update via
Initialization	RESET	-
Cyclic processing	Sampling timer	-
Alarm-driven processing	Alarms (e.g. hardware interrupts, diagnostic interrupts, etc.)	-
Process image partition	-	PROFINET cycle
Exception handling and diagnostics	System interrupts	-
Communication, input/output	I/O interrupts	-
Service	Operator control & monitoring	-
User program	-	-
Utility programs	-	-

Initialization

Initialization is triggered on activation of the power supply, or by a pulse signal of the RESET button. The initialization routine must prepare the hardware and software in such a way that the system is able to enter the standard operating state (RUN operating state).

Cyclic processing (RUN operating state)**CPU551**

The Task Manager ensures cyclic processing of the functions assigned to the different tasks. Cyclic tasks are executed at a ratio to the power of 2 to each other

$T_a = T_0 \times 2^b$ with T_0 as basic sampling time,
b defines the sampling time value with $0 \leq b \leq 15$
a numbers the sampling times with $1 \leq a \leq 5$.

CPU555

$T_a = T_0 \times Y$ with T_0 as basic sampling time,
 $Y = 1/4, 1/3, 1/2$, integer multiples of T_0
a numbers the sampling times

Example:

The following sampling times are possible with a basic sampling time of 1 ms:

- CPU551: 1 ms, 2 ms, 8 ms, 32 ms and 128 ms
- CPU555: 0.25 ms, 0.33 ms, 0.5 ms, 1 ms, 2 ms, 3 ms to 65535 ms

The basic sampling time is defined for each CPU module in the configuration phase using the HW Config program section of SIMATIC Manager. The sampling times of the tasks executed on the respective CPU module are also configured in this phase.

In order to prevent bottlenecks, the tasks are phase shifted and started at the basic clock cycle, so the start of a second, lower-priority task is flagged with the basic clock cycle in each case. All low-priority tasks are taken into account, which means that no more than one low-priority sampling time can be active in the basic cycle clock. (see section "Significance of the processor utilization (Page 58)"). Task priorities are reduced in proportion with rising sampling times.

The Task Manager is started with the clock cycle of the basic sampling time set at the sampling timer. This calculates the second task, task T_n to be started in addition to T_1 (T_n from $\{T_2...T_5\}$). If the task to be started has a lower priority than an interrupted task, its start is buffered and the interrupted task is continued. Otherwise, the calculated task is started. The status of the interrupted task is written to a task-specific data area that enables further processing as soon as no higher-priority task is pending.

The time component required by the operating system itself is not taken into account in the representation. A precise diagram would show that the actual start time of the task is shifted by these amounts.

Alarm-driven processing

In addition to cyclic processing, the operating system also manages tasks that are started by non-cyclic interrupts, in particular by hardware interrupts. Possible alarm sources:

- Software interrupts
- CPU timer interrupts
- PROFINET hardware interrupts
- PROFINET diagnostic interrupts
- PROFINET IO clock cycle

The priority of alarm tasks is specified based on the configuration in HW Config (I1 > I2...> I8). The application programmer uses HW Config to configure the alarm sources that are necessary for the application, as well as their processing in the alarm-driven tasks.

Alarm tasks always have higher priority than cyclic tasks on a CPU551.

On a CPU555, you also specify the priority of the alarm when you select the hardware interrupt. A smaller number means higher priority here. Higher-priority tasks are run before lower-priority tasks and can interrupt a lower-priority task. The following priority is defined in the context of the cyclic tasks:

Priority	CPU551		CPU555	
	Alarm	Cyclic	Alarm	Cyclic
0 (highest)	I1		I1	
1	I2		I2	
2	I3			T1
3	I4		I3	
4	I5			T2
5	I6		I4	
6	I7			T3
7	I8		I5	
8		T1		T4
9		T2	I6	
10		T3		T5
11		T4	I7	
12 (lowest)		T5	I8	

Process image (system mode)

Before you start task processing, you first have to determine whether or not a corresponding process image needs to be updated. If so, call the system mode of the function blocks before you start the task (see section "Significance and application options of the process image (Page 50)"). The update is related to:

- Digital input/output data, e.g. the status images for controller enable signals and the position of limit switches.
- Analog input/output signals, e.g. values for temperature, speed, etc.

Note

The system mode is initiated for both tasks to be started prior to standard mode processing (see section "Significance and application options of the process image (Page 50)").

Process image partition

You define the process image partition with the address assignment. The system updates the process image partition automatically through the CPU555. To read values of the process image input, use the CRV_P function block. To write values into the process image partition of the outputs, use the CTV_P function block.

Error differentiation

SIMATIC TDC differentiates between initialization error events and error events in standard operation.

Initialization errors (INIT operating state) prevent activation of the start enable signal for the system (transition to the RUN operating state).

For errors in standard operation (RUN operating state), you must decide whether to allow further operation, or whether it is necessary to abort processing.

The system informs you of its status, in particular of its error states by means of the seven-segment display on the CPU module.

Detailed error information is stored in the OS error fields that enable precise error analysis.

These data can be read and edited using the service utility.

For more information on the significance of error signals, and corresponding remedies, refer to "D7-SYS, Help on events" in the Online Help.

Communication

Communication handles the entire I/O data traffic between the hardware, corresponding software components, and user interfaces. The interfaces and their parameterization are configured in the user program using CFC.

Service utility

The service utility represents the central interface of the CPU modules and is instrumental for commissioning, diagnostics, and troubleshooting.

As the processing time of the service utility is undefined, the task linked to the utility and tasks with lower priority may be inhibited. This was implemented by the assignment of a maximum processing time to the service within its cycle (maximum of one basic clock cycle T0).

The user interface is formed by service devices that are controlled by means of the communication software.

User program

The user program realizes the technological tasks on the target hardware. It is created on the programming device in CFC programming language with the help of utility programs such as HW Config, CFC editor, CFC compiler, linker/locator, and memory module drivers.

The CFC source code of the user program is converted into data structures by the CFC compiler and provided on the target hardware for processing by the operating system.

Utility programs

This is a generic term for basic functions of the operating system. These include wake-up functions, functions for handling the CPU display, as well as special test and interrupt routines for handling system errors.

2.1.8.3 Service utility

The service utility provides a pool of information functions that enable you to access local system data of the processor. The service utility is intended to provide help in commissioning and testing.

Commissioning applications

The utility enables you to visualize and/or edit configuration data (setpoints/actual values) and to optimize the configuration (e.g. by editing interconnections, controller times, etc.).

Test applications

The tool supports you in locating sources of system malfunction (crash, startup problems) and disturbance which originate from the CPU module.

All activities of the service utility are controlled by means of tasks received via "its" data interface (corresponding to the parameter assignments for the service function block I/O).

All HMI devices capable of processing the task/response language of the utility can be used for the service utility. In the SIMATIC TDC product range, these are the CFC programs (tools) in test mode.

The service utility is made available with the SER function block. This function block ensures that none of the messages are lost.

Job processing

The service utility differentiates between cyclic and non-cyclic tasks. A non-cyclic task is completed after its response frame has been transmitted.

A cyclic task remains active until explicitly canceled by a reset or new task. A task always comprises at least one response frame.

Note

The service utility is not capable of processing several tasks in the same session. This means that the next task is not processed until a response frame was received to the previous task.

System load, response times

The actual service utility processing is realized in a 32 ms sampling time slice (the next sampling time below 35 ms is selected; the sampling times specified at the SER blocks are insignificant for processing). The service blocks used in the cyclic task are provided a certain computing time slice that may not exceed the maximum basic clock cycle T_0 . The ratio between the basic clock cycle T_0 and the task used determines CPU performance and therefore system load.

Example 1:

Basic clock cycle $T_0 = 1$ ms; selected sampling time = 32 ms. A time slice of 1 ms is reserved for the service utility in all 32 ms cycles. System load is therefore calculated based on the equation

$$1 \text{ ms} / 32 \text{ ms} = 0.03125 = 3.125\%$$

Example 2:

Basic clock cycle $T_0 = 2$ ms; selected sampling time = 16 ms. A time slice of 2 ms is reserved for the service utility in all 16 ms cycles. System load is therefore calculated based on the equation

$$2 \text{ ms} / 16 \text{ ms} = 0.125 = 12.5 \%$$

The available computing time is used by all service blocks to the same extent, i.e. all SER blocks will be executed once, provided sufficient time is available. An SER block processes only one task per clock cycle. For cyclic tasks, only one response frame is received in each cycle. The advantage of this mode of operation is that it is possible to generate responses with constant bus cycle time for the cyclic tasks.

Remainders of reserved computing time slices not used to the full extent are made available to the system, e.g. if no task execution is pending.

For multi-processing configuration with simultaneous access to system resources that are only available once (e.g. change memory of the memory module), resources are assigned to the first component requesting these. All other requests are rejected and trigger the output of an error message ("Resource in use") via the data interface no later than 1 second following the request.

Behavior in the event of a fault

The system goes into stop mode in the event of a fault (exception), i.e. at initialization or online errors. This situation sets special conditions for the service utility, which means that calculations are no longer performed in a cyclic task and, instead, run in a continuous mode that is started by an exception handler. There are incidents in which the service utility cannot be reached, for example on a CP. In order to enable system diagnostics in spite of this scenario, the CPU's internal diagnostics interface will be connected. With CPU551, the DUST1 protocol runs over the interface X1 (see section "Operating states of a CPU module (Page 42)").

With CPU555, diagnostics is performed via one of the PROFINET interfaces (X1, X2, X3).

2.2 Function description and user instructions

2.2.1 Fatal system error "H"

If a fatal system error occurs, processing (initialization or normal operation) is interrupted and the system goes into the stop mode. The error cause is then made available for diagnostics services.

Note

Best practice in this case is to analyze the INIT_ERR and SYS_ERR system error fields before you analyze fatal system errors. The error entries in these fields (particularly hardware (monitoring) errors) possibly indicate the cause of a fatal system error.

A SAVE area is set in the upper area of local RAM in all CPU modules. This area is not erased by an initialization if the status of the RAM copy is appropriate. An error buffer is set up in this SAVE area and contains the error error report which consists of several messages.

The error buffer consists of a management element and a ring buffer for storing the error messages. The ring buffer is implemented as overwrite buffer, which means that the oldest message will be overwritten with a new message on buffer overflow.

There are two different types of error messages. A long message is output for non-maskable interrupts (NMI) and a short one for power OFF.

The communication service utility is available (even if not configured) for analyzing fatal system errors. You can access this utility via the local diagnostics interface by pressing the acknowledge button. The error causes are output in plain text with the help of the service utility.

The error cause that is specified under the ID and supplementary ID is of particular importance. This information is output if a function block is being processed at the instant of system error. The service also displays the results of recent bus accesses, which are of importance if a bus access has caused the error. Moreover, is also displays all processor registers for precise fault analysis by system experts.

Causes of fatal errors

A fatal system error may have the following causes (ID codes). The supplementary ID describes the error cause in more detail.

Identifier	Supplementary ID (precise definition)
CPU	<ul style="list-style-type: none"> • Exceptional state of the CPU Internal fault Reserved instruction Unknown syscall unaligned instruction fetch (jump to address with value that cannot be divided by four) user access to kernel space unaligned load/store to coprocessor 0/2/3 unaligned load/store to L/C bus address space break 6/7 not in div/mul context unknown break value reserved exception Task in infinite loop
FPU	<ul style="list-style-type: none"> • Exceptional state of the FPU fpu fault at non-fpu instruction illegal fpu sub opcode operation on NaNs add/sub/division of infinities mul of infinity and 0
TLB	<ul style="list-style-type: none"> • Exceptional state of the TLB TLB modified exception TLB read/write miss (access to invalid address) UTLB miss (access to invalid address)
TIME	<ul style="list-style-type: none"> • Basic sampling time failure
OFF	<ul style="list-style-type: none"> • Power down Power down/reset in normal operation Power down/reset in STOP mode (following other exception)

If the CPU has no tasks to process during normal operation, it processes the background task.

The following functions are simultaneously available as background task:

- Online test mode
- A service utility

The online test mode is normally processed in the background after initialization was successfully completed. However, if the acknowledge button is pressed at the end of initialization, only the service communication service utility will be activated.

Errors in background processing are saved in the UEB element of the SYS_ERR error field.

Online test mode

The online test mode includes actions such as battery tests, or memory module checksum tests. The memory module checksum routine calculates the memory module checksum and compares the result with the value calculated offline on the programming device, which is stored in the memory module. If a memory module checksum error is detected in the online test mode, you can eliminate this error by generating the memory module again. In the case of battery test errors, you can replace the battery.

2.3 System chart @SIMD

Overview

System chart @SIMD is a CFC that is made available by default to users to enable standard diagnostics of the hardware and system software. The chart consists of the chart partitions A and B.

Program structure

The system chart is organized in the following parts

- Identifying acknowledgment Acknowledgment of the error display
- Evaluating components Determination of the component that reported an error
- Display Output the identified error

System chart @SIMD		
		Function block names
	Identifying acknowledgment	
	Pushbutton	ACK Acknowledge
	Service intervention	ACK
	Evaluating components	
	First error field	FER First error
	Communication error field	CER communication error
	Task Manager error field	TER Task management error
	Hardware failure monitoring error field	HER HW error
	User error field	UER User error
	Evaluate errors	DER display error
	Display	
	Output seven-segment display	DST display status
	Output diagnostics LED	DST
	Output status word SIMS, status bit SIMD	SIMS, SIMD

Description

The operating system monitors the hardware and the system software. If the monitoring function identifies an error, it flags this by setting the appropriate bits (flags) in the system error field.

The system chart @SIMD makes these flags available to the user. An output is displayed on the seven-segment display of the CPU module if a flag of a component was set.

If several messages are generated for the seven-segment display, the highest-priority message is output.

Table 2- 10 Important error priorities for the message display

Error description	Error display	Priority
Communication error	C	High
Task Manager error	E	
PROFINET IO error	P	
Monitoring error	b	
User error	A	
No error pending	CPU number	Low

For details, see "SIMATIC TDC hardware (<http://support.automation.siemens.com/WW/view/de/8776697/0/en>)".

Resetting the flags

The flags of the displayed error is reset and the next priority error code is displayed when you press the acknowledge button on the CPU module or when an acknowledge signal is output by a service device. If no errors pending, the CPU number is displayed on the seven-segment display as the lowest priority message. A flashing error message identifies the error message as the first one that has occurred.

Functional principle

The flow diagram (see "Flow diagram") represents the global program runtime of the system chart and consists of three functional components

- Identifying acknowledgment
- Evaluating components
- Display.

Identifying acknowledgment

The acknowledge signal is a pulse which is derived from the pushbutton status read from the ASI function block, or as result of a service intervention at I/O ACK000.I (set from 1 to 0). Priority-controlled error fields and therefore their display are acknowledged using this pulse. Output of error codes "C" and "E" can be suppressed by changing the ACK050.I I/O from 0 to 1.

Evaluating components

The components are evaluated using the function blocks SYF1 and SYF4. The corresponding numbers of the error fields are documented in the function block description. An error field can only be acknowledged if an error was identified and displayed for the particular component.

Evaluating the first error field

The first error field evaluation determines which error entry was the first to be identified by the system. The error entry in the first error field flashes on the seven-segment display.

All components are evaluated successively based on their priority. The communication error field cannot be acknowledged, as it is necessary to edit the configuration in order to clear this error. The CPU may be subject to higher load at system startup. Task management errors are acknowledged automatically during system startup using a count logic function.

Control using priority logic

A priority logic ensures that only the top priority component is displayed. The lowest-priority component returns a bit signal that toggles the CPU number display to the error display (UER070.Q). The error display flashes if the top priority error component is additionally entered in the first error field.

An acknowledge pulse only resets a single error status of a component and its display.

Note

The error source is still present after a displayed error has been acknowledged. Before an error can be eliminated, it is necessary to clear the cause of error.

Display

If no errors are pending, the station outputs its own processor number to the seven-segment display. A corresponding error code is output if a component signals an error.

Flow chart

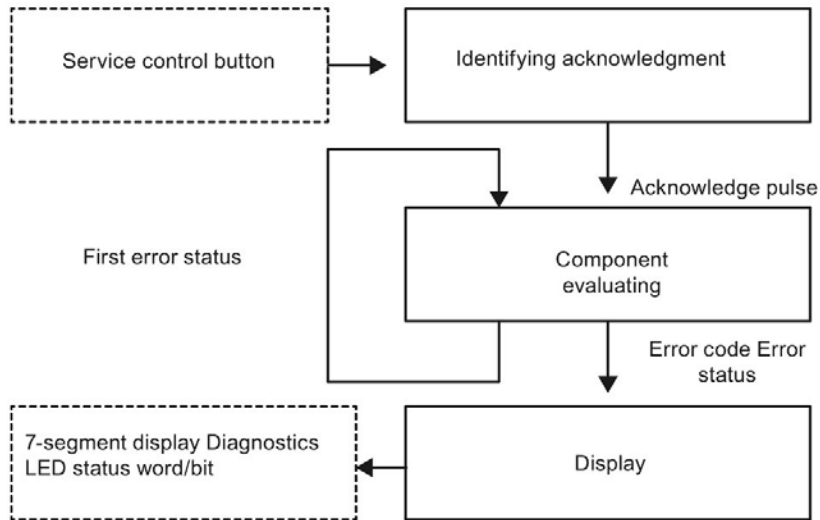


Figure 2-11 Flow chart

Interfaces

The acknowledge button of the CPU module, or an optional acknowledgment via the service interface, can be used as external input of the system chart. The seven-segment display of the CPU module are available as an external output for the user display.

The two connections SIMS.QS and SIMD.Q can be evaluated for error handling in the user program. The error outputs of the individual components are combined to form an error status word by means of the SIMS function block. The SIMD.Q output represents a general error status.

The error status word at the SIMS.QS block connection has the following bit assignment:

Table 2- 11 Bit assignment of function block I/O SIMS.QS

Bit	Meaning
1	n.c.
2	n.c.
3	n.c.
4	Task Manager error
5	n.c.
6	Hardware failure
7	Communication error
8	n.c.
9	n.c.
10	n.c.
11	User-generated error ID
12	n.c.
13	n.c.
14	n.c.
15	n.c.
16	n.c.

Configuring communication

3.1 Introduction

3.1.1 Communication basics

General

Communication enables the exchange of information between different systems and devices.

Communication requirements:

- Configuration of a communication service and of a link
- A communication interface must be available

Communication service

The communication service defines the information content (e.g. process data) for communication.

Coupling

The coupling defines the hardware (e.g. CP50M1) and the transmission protocol (e.g. PROFIBUS DP) for communication.

Couplings and communication interfaces

The application and communication capabilities of the partner determine the communication interface and the coupling.

Communication connections per CPU

Each CPU can manage a maximum of 4000 connections (communication, service and message channels).

3.1.2 Overview of communication utilities

General

The communication interfaces can be used to transmit diverse information such as process data and messages.

The communication utilities determine to information to be transmitted. The communication utilities are defined by configuring the communication modules.

Communication utilities

Table 3- 1 Overview of communication utilities

Communication utility	Description	Communication blocks to be configured
Alarm logging	Configuring warning and alarm systems	Special message blocks: @MSC, MER..., MSI...
Process data	Transmission of process data (setpoints and actual values)	Transmit and receive blocks: CTV, CRV, CCC4, CDC4
Service	Diagnostics and analysis of CPU programs / CFC	Service function block: SER
Time synchronization	Time synchronization of all CPUs used (e.g. to compare messages with time stamp).	Special function blocks: RTC...
Data logging (Trace)	Tracing process variables	@TCP, TR...
S7 communication	Operator control and monitoring of CPU programs / CFC	Communication function block: S7OS

3.1.3 Overview of couplings

General

Couplings are configured in the CFC application using the central coupling blocks.

Local CPU coupling

Application for communication partners	<ul style="list-style-type: none"> Internal CPU function for testing senders/receivers
Necessary hardware	<ul style="list-style-type: none"> CPU
Communication utilities	<ul style="list-style-type: none"> Process data
Central coupling block	<ul style="list-style-type: none"> @LOCAL
Features	<ul style="list-style-type: none"> Internal memory buffer module in SIMATIC TDC

Direct CPU-CPU coupling

Application for communication partners	<ul style="list-style-type: none"> CPU-CPU communication for larger data volumes (as an alternative to \$ signals via VME bus including the P0 connector)
Necessary hardware	<ul style="list-style-type: none"> CPU
Communication utilities	<ul style="list-style-type: none"> Process data
Central coupling block	<ul style="list-style-type: none"> @LOCAL
Features	<ul style="list-style-type: none"> Internal memory buffer module in SIMATIC TDC

See also Direct CPU-CPU coupling (Page 103).

Buffer memory coupling

Application for communication partners	<ul style="list-style-type: none"> CPU-CPU communication for larger data volumes as an alternative to \$ signals
Necessary hardware	<ul style="list-style-type: none"> Memory buffer module (CP50M1 / CP51M1 or CP53M0)
Communication utilities	<ul style="list-style-type: none"> Process data
Central coupling block	<ul style="list-style-type: none"> @GLOB
Features	<ul style="list-style-type: none"> Internal memory buffer module in SIMATIC TDC

Subrack coupling SIMATIC TDC

Application for communication partners	<ul style="list-style-type: none"> • SIMATIC TDC
Necessary hardware	Communication modules for the master interface: <ul style="list-style-type: none"> • CP52M0 (GDM-Memory) • CP52I0 (GDM-Interface)
Necessary hardware	Communication module for the slave interface: <ul style="list-style-type: none"> • CP52A0
Communication utilities	<ul style="list-style-type: none"> • Process data, message system, trace
Central coupling block	<ul style="list-style-type: none"> • @SRACK
Features	<ul style="list-style-type: none"> • Fiber-optic cable • Parallel coupling of up to 44 SIMATIC TDC racks • All racks can be synchronized • Uniform system clock possible • Fast • The maximum distance between 2 racks is 200 m • You may always disable the rack

Rack coupling for TDC with SIMADYN D

Application for communication partners	SIMATIC TDC with SIMADYN D
Necessary hardware	Communication module for SIMATIC TDC: CP53M0
Necessary hardware	Communication module for SIMADYN D: CS12/CS13/CS14/CS22
Communication utilities	Process data
Coupling central block	@CS1 (master mode) @CS2 (slave mode)
Features	Fiber-optic cable Parallel coupling of SIMATIC TDC with SIMADYN D Parallel coupling of up to 3 SIMATIC TDC racks All racks can be synchronized Uniform system clock possible Fast The maximum distance between 2 racks is 200 m You may always disable the rack

TCP/IP coupling

Application for communication partners	<ul style="list-style-type: none"> • SIMATIC TDC • SIMATIC S5/S7 • Third-party systems • WinCC
Required hardware	<ul style="list-style-type: none"> • CPU555 or • CP51M1 communication module
Communication utilities	<ul style="list-style-type: none"> • Process data and message system • Service • S7 communication
Central coupling block	<ul style="list-style-type: none"> • @TCPIP • @PNIO
Features	<ul style="list-style-type: none"> • Standardized bus to Ethernet (IEEE 802.3) • Baud rate: 10 or 100 Mbps (autosensing)

PROFIBUS DP

Application for communication partners	<ul style="list-style-type: none"> • SIMATIC S5/S7 • ET200 • SIMATIC TDC • Certified third-party devices
Required hardware	<ul style="list-style-type: none"> • CP50M1 communication module
Communication utilities	<ul style="list-style-type: none"> • Process data • Parameter processing
Central coupling block	<ul style="list-style-type: none"> • @PRODP
Features	<ul style="list-style-type: none"> • Standardized multi-master bus for communication between SIMATIC TDC and up to 123 communication partners • Master-slave principle (CP50M1) • PROFIBUS standard to EN 50170 • Fast • Max. 12 Mbps • The maximum user data length is 244 bytes

MPI

Application for communication partners	<ul style="list-style-type: none">• CFC• WinCC• SIMATIC OPs
Necessary hardware	<ul style="list-style-type: none">• CP50M1 communication module
Communication utilities	<ul style="list-style-type: none">• Service• S7 communication
Central coupling block	<ul style="list-style-type: none">• @MPI
Features	<ul style="list-style-type: none">• Multi-master bus with up to 126 nodes• 187.5 Kbaud / 1.5 Mbps• Standard in SIMATIC S7

PROFINET

Application for communication partners	<ul style="list-style-type: none">• PROFINET devices• Distributed I/O
Necessary hardware	<ul style="list-style-type: none">• CPU555
Communication utilities	<ul style="list-style-type: none">• Process data and message system• Service• S7 communication
Central coupling block	<ul style="list-style-type: none">• @PNIO
Features	<ul style="list-style-type: none">• Standardized bus to Ethernet (IEEE 802.3)• Data transmission speed: 10 or 100 Mbps (autosensing)

3.1.4 Communication block I/O

3.1.4.1 Initialization connection CTS

Communication blocks which access a data interface have a CTS I/O.

Data specified at the initialization I/O

Data specified at the CTS I/O:

1. The name assigned to the communication module

Naming conventions:

- The name string has a length of 1 - 6 characters
- Character 1: A - Z
- Characters 2 - 6: A - Z, 0 - 9, _

2. Connector for a data interface that is installed on a CP50M1 or CP51M1 communication module

Rules for the connector name:

- A dot "." is appended to the module name
- The name string appended to the dot "." has a length of 3 characters
- "X01", "X02", or "X03"

Connector of the data interface for the CPU555.

Rules for the connector name:

- Enter "." after module length
- The name string appended to the dot "." has a length of 3 characters
- "P0" (P0 connector)
"PN" (PROFINET)

Example of data specified at the CTS

Example of a rack configuration with CPU551:

Slot	Module	Configured module name in HW Config	Possible information at CTS I/O
S01	CPU551	"D01P01"	"D01P01"
S03	CP50M1	"COUPLE"	"COUPLE"
S04	CP50M1	"COMM1"	"COMM1.X01" "COMM1.X02"
S06	CP51M1	"TCPIP"	"TCPIP.X01"

Example of a rack configuration with CPU555:

Slot	Module	Configured module name in HW Config	Possible information at CTS I/O
S01	CPU555	"D01P01"	"D01P01"
			"D04P04.P0"
			"D01P01.PN"
S03	CP50M1	"COUPLE"	"COUPLE"
S04	CP50M1	"COMM1"	"COMM1.X01" "COMM1.X02"
S06	CP51M1	"TCPIP"	"TCPIP.X01"

3.1.4.2 Address connections AT, AR, and US

General

Communication blocks that are allowed to access a data interface have an address connection.

Address connection types

We distinguish between three address connection types based on the particular block type:

- AT connection: Available for transmitters
- AR connection: Available for receivers
- US connection: Available for function blocks that process transmit and receive channels

Possible address connection data

The data specified at the address connection are independent of the AT, AR, or US types. Possible data specified:

- "Channel name"
- "Channel name.Address Level 1"
- "Channel name.Address level 1.Address level 2"

Channel name

- The channel name addresses a channel on a data interface.
- Transmitters and receivers accessing a data interface with the same channel name communicate.
- The channel name consists of up to eight ASCII characters, excluding the "dot" and "@".

Note

A check of redundant channel name assignments is not performed. The configuration engineer must assign unique channel names at the AT, AR, or US connections for each transmitter/receiver on a data interface. If this condition is not met, the

- transmitter/receivers are possibly subject to uncoordinated multiple usage
- the transmitter/receiver will possibly log off with a communication error.

Exceptions:

- Several transmitters are allowed in the "Select" transmission mode.
 - Several receivers are allowed in the "Multiple" transmission mode.
-

Address levels

- The program provides address levels 1 and 2.
- The address level must be defined to enable data transmission at certain couplings such as PROFIBUS and Industrial Ethernet. No address levels are specified for rack couplings, for example.
- Address level 1 has a maximum length of 14 and address level 2 a maximum of 20 characters.
- Address level 2 is not supported by the CTV_P and CRV_P blocks.
- Significance and contents of the address levels are described for the respective coupling.

3.1.4.3 Transmission mode, MOD connection

Overview

Five different transmission modes are available to suit the various communication requirements:

- Handshake
- Refresh
- Select
- Multiple
- Image

Selecting the transmission mode

The transmission mode is specified at the MOD connection of the corresponding transmitter or receiver.

"Handshake" transmission mode

The "Handshake" transmission mode is used

- if information may not be lost by data being overwritten, and
- if exactly one receiver has to be available for each transmitter.

"Handshake" describes sequential channel processing. The transmitter first saves a new data set in the channel after the receiver has acknowledged receipt of the recent data set. A user data buffer is provided for data exchange.

The transmitter enters the user data in the channel in a single operating cycle and the receiver reads the data in a work cycle.

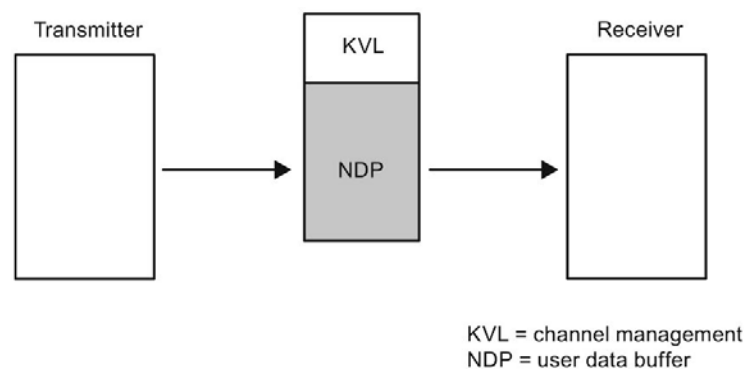


Figure 3-1 Data transmission in "Handshake" mode

"Refresh" transmission mode

The "Refresh" transmission mode is used

- if the latest data is always to be made available to a receiver and
- if exactly one receiver has to be available for each transmitter.

"Refresh" is an overwriting data exchange. The transmitter always saves the latest data set in the channel without the receiver acknowledging the receipt of the last data set. Two user data buffers that are made available for data exchange are used as alternating buffer system. The transmitter reports the buffer in which the latest data are stored.

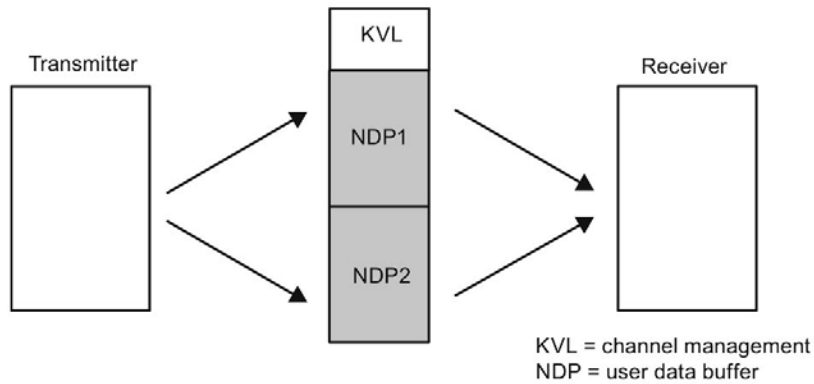


Figure 3-2 Data transmission in "Refresh" mode

"Select" transmission mode

The "Select" transmission mode is used

- if information may not be lost by data being overwritten, and
- if any number of transmitters can be made available for one receiver

"Select" defines a sequential channel processing mode. After the receiver has acknowledged the receipt of the last data set, the transmitter writes a new data set to the channel. A user data buffer is provided for data exchange. A channel management function controls data traffic.

All configured transmitters use the same user data buffer. There is no fixed transmission order for the transmitters, i.e., the first one always sends first. In order to achieve controlled data transmission, a "1" may only be set for one transmitter at the EN connection.

The transmitters must be configured for a shorter sampling time than the receiver.

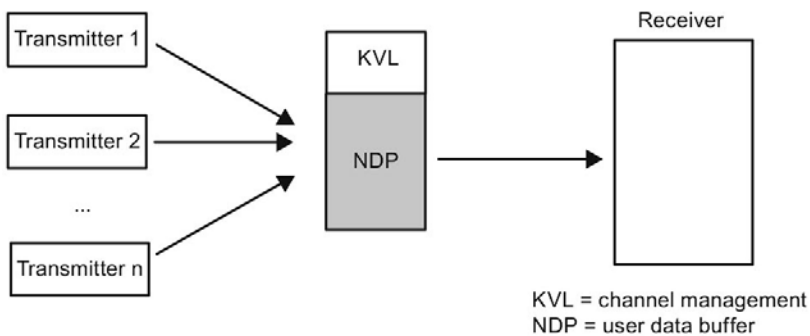


Figure 3-3 "Select" data transmission mode

"Multiple" data transmission mode

The "Multiple" data transmission mode is used

- if receivers are to always be provided with the latest data
- if as many receivers as required are available for each transmitter.

"Multiple" is an overwriting data exchange mode. The transmitter always saves the latest data set in the channel without the receiver acknowledging the receipt of the last data set.

If a transmitter overwrites a buffer, from which a receiver is presently reading, then the receiver rejects the data which were last received. Receive mode is repeated in the next operating cycle.

Two user data buffers that are made available for data exchange are used as alternating buffer system. The transmitter reports the buffer that contains the latest data.

The receivers must be configured in the same or shorter sampling time than the transmitter (the receivers must therefore operate faster).

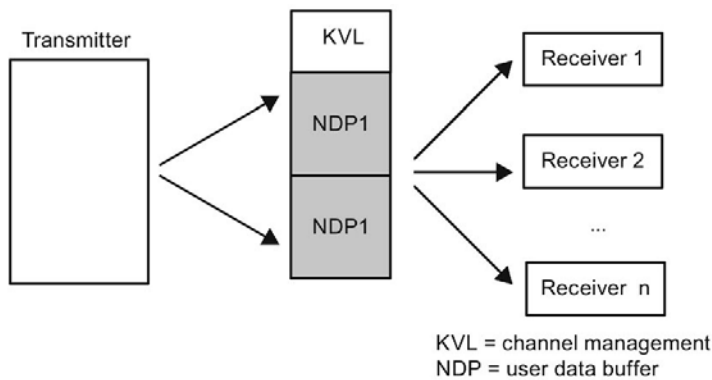


Figure 3-4 "Multiple" data transmission mode

"Image" data transmission mode

Note

FM 458-1 DP only

The "Image" transfer mode is only allowed for the PROFIBUS DP interface of the FM 458-1 DP application module.

The "Image" transfer mode is used for communication via the PROFIBUS DP interface:

- if all receivers configured in a task must be provided data that originates from the same DP cycle,
- if all transmitters configured in the same task should transmit their data to the DP slaves in the same DP cycle.

For this purpose, transmitter and receiver FBs are synchronized automatically within a task in order to return consistent data. They form what is referred to as a "**consistency group**". All receiver FBs in such a consistency group fetch their user data from a common alternating buffer, while all of the transmitter FBs save their data to such a buffer.

"Image" is an overwriting data exchange mode (refer to refresh). The two user data buffers for data exchange re used as alternating buffer system.

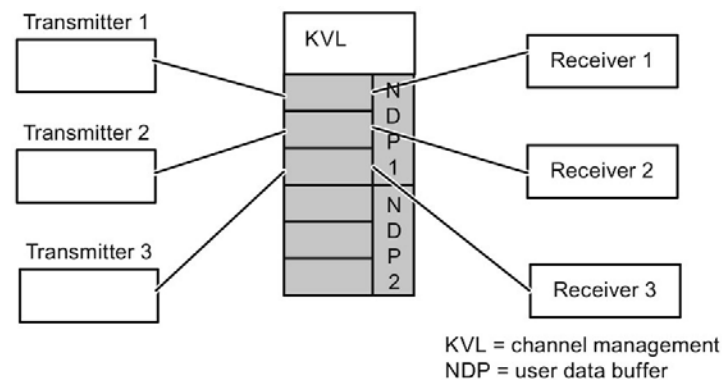


Figure 3-5 "Image" data transmission mode

3.1.4.4 Firmware status, ECL, ECO connection

General

Central coupling blocks that communicate with a firmware (e. g. @PRODP) have ECL and ECO outputs.

Function

The ECL and ECO outputs indicate the status of the corresponding firmware:

- ECL=0 and ECO=0:

The firmware is in error-free state.

- ECL=0 and ECO>0:

The firmware error pending can be eliminated by the configuration engineer or user. The cause of the error is described in the chapters related to the individual couplings.

- ECL>0 and ECO>0:

Irreparable firmware error

3.1.4.5 Status display, output YTS

General

The block outputs an error code, or the current transmission status at its YTS output.

Displayed errors

- Real (severe) runtime errors
- Configuration errors that are identified during system initialization and indicated on the seven-segment display of the CPU by a flashing "C".
- Temporary status displays and alarms

Error diagnostics

The value at output YTS can be read as decimal number using CFC.

For more information on the significance of the decimal number, refer to the "SIMADYN D, Help on events" Online Help (press F1 while working in the CFC and select "Help on events" in the "CFC for SIMATIC TDC" section).

3.1.5 Function principle of couplings

General

A coupling functions as follows:

- CPUs exchange data with a coupling module via the backplane bus.
- The firmware on the coupling module "reorganizes" and "packs" the data of serial couplings (e.g. TCP/IP) for compliance with the specified frame structure and protocol.
- The data will not be conditioned if the communication partner is also a SIMATIC TDC (rack coupling, buffer memory coupling).

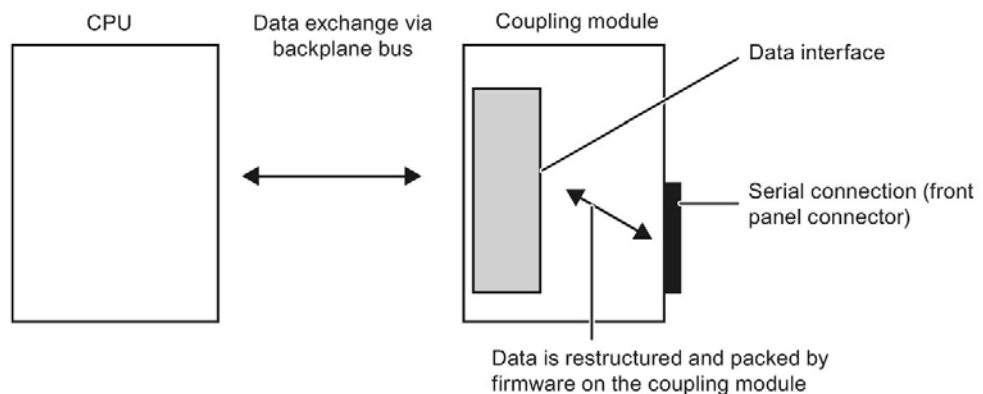


Figure 3-6 Data exchange between the CPU and coupling module

Access to the data interface

The data interfaces can be used by all CPUs in a rack, as these interfaces are located on external coupling modules and not locally on a CPU. Operation of the CPU and coupling module on the same bus connection is conditional for using a data interface.

Note

For local CPU coupling, the data interface is set up in local RAM of the CPU. This data interface cannot be accessed by other CPUs of a rack and is only available for CPU on which it was configured.

Basic initialization

The basic initialization of the coupling module is always performed at system startup.

The first CPU performs the following tasks:

- checks as to whether the coupling module can be "addressed"
- formatting of the data interface

Configuration of the coupling module

The desired coupling module is configured in HW Config. Explicit configuration steps do not have to be performed for basic initialization of a coupling.

3.1.5.1 Central coupling blocks

Function of the central coupling blocks

The central coupling blocks provide the following functions:

- Initialization:
 - Copying the configured initialization data (configured for the initialization connections) to the data interface
 - Verification of the error-free state of the data interface.
- Enable:
 - After initialization by the central coupling blocks and the coupling module firmware, the central coupling block enables the coupling for all transmitters and receivers in the same rack. Data transmission can now start
 - For timing reasons, a coupling is not enabled in the RUN operating state until several sampling cycles have been completed.
- Monitoring:
 - The central coupling blocks return information at their outputs on the coupling and firmware states.

Configuring the central coupling blocks

The following aspects must be taken into account when configuring the blocks:

- Exactly one central coupling block must be configured for each coupling.
- All central coupling blocks can be configured on a single CPU of a rack, or on several CPUs of a rack.
 - The configuration of all central coupling blocks on a single CPU simplifies diagnostics, for example.
 - Central coupling blocks have no transmitter or receiver functionality.
 - A sampling time of 32 ms TA 256 ms must be set for all central coupling blocks

Error

The central coupling block generates an error message in the communication error field and no longer processes the coupling module

- if a central coupling block detects an initialization error
- if the firmware does not respond, or if malfunction of the firmware is detected
- if the central coupling block is running on the incorrect communication module.

3.1.5.2 Transmitters and receivers

General

Transmitters and receivers are in particular:

- Function blocks with read and/or write access to the data interface of a coupling

Component of the communication utility that uses the coupling. Examples of transmitters:

- Message evaluation FB MSI:

Copies messages from the message buffer to a data interface

Process data transmit block CTV. Examples of receivers:

- Process data receive block CRV

Definitions at the connections

A coupling type must not be specified at the block inputs of the transmitter and receiver, as transmitters and receivers do not distinguish between the various couplings.

Connections of the transmitters/receivers

- CTS connection for specifying the coupling module
- Address connection AR, AT, or US for specifying channel names and coupling-specific addresses

Synchronizing transmitters and receivers

Before transmitters and receivers are able to exchange data, they first need to exchange their "identification" data and get "synchronized":

- This identification is based on the configuration data set at the CTS, AT, AR, or US connections.
- Synchronization is only if the following conditions are met:
 - The transmitter identifies its partner as receiver (or vice versa)
 - Matching length of reserved data areas
 - Compatible user data structure
 - The same transmission mode is set at the partners (data specified at the MOD connection for transmitters/receivers).

If one of these conditions is not met, the synchronizing transmitter/receiver logs off with a communications error.

Note

In normal mode, transmission and receiving does not start at the first cycle, but rather after the data interface of the module configured at input CTS has been enabled and the corresponding coupling partner has been identified (cf. connection output YEV). This may last for the duration of several sampling times.

3.1.5.3 Compatible user data structures

General

The user data structures contain information on the structure of user data to be transmitted:

- Specification with regard to the position and types of corresponding user data

The structure of user data of the transmitter and receiver must be identical to enable data exchange between the transmitter and receiver.

Data types

The following standardized data types are used:

Table 3- 2 Standardized data types

Standardized data type	SIMATIC TDC data type	Length in bytes
Integer 16	Integer	2
Integer 32	Double Integer	4
Unsigned 8	Bool, Byte	1
Unsigned 16	Word	2
Unsigned 32	Double Word	4
Floating Point	Real, SDTIME	4
Octet String	-	1
Time and Date	-	6

Note

The SIMATIC TDC connection types (e.g. SDTIME) are not used as data types, as the coupling partner does not necessarily have to be a SIMATIC TDC function block.

- **Octet String**

An octet string is an unstructured data type that does not appear at the block I/O (refer to the chapter "Channel marshalling blocks CCC4 and CDC4").

- **Time and Date**

Combined data type for the time that does not appear at the block I/O (refer to chapter "Communication utility message system").

Range of values

- **Octets 1 and 2:**

Specify the date relative to 1.1.1984 an.

Resolution = 1 day

$0 \text{ days} \leq d \leq 65535 \text{ days}$

- **Octets 3 to 6:**

Specify the time between 00:00 h and 24:00 h.

Resolution = 1 ms

$0 \text{ ms} \leq x \leq 86400000 \text{ ms}$

The first four bits in octet six are unused

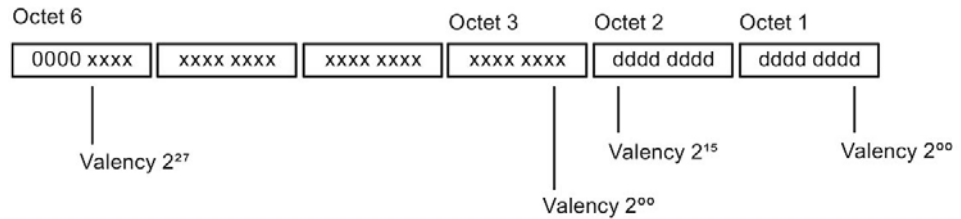


Figure 3-7 Time and Date

3.1.5.4 Number of coupling modules in a rack

Overview

The number of CS coupling modules (CP51M1 and CP52A0) is restricted by two system limits:

- Rack size
The largest rack in the SIMATIC TDC system is equipped with 21 slots. Of these, 20 slots remain available in theory as at least one CPU must be inserted in the rack.
- Available address space
However, this limit is hardly reached in practical life. 144 MB of address space is available on the bus for CP modules.

Occupied address space

CP modules always occupy a constant address space on the backplane bus.

- **Example**
CP52A0 always occupies 2 MB on the bus.

Module type	Occupied address space
CP52A0	2 MB
CP50M1	1 MB, or 9 MB as buffer memory
CP51M1	1 MB, or 9 MB as buffer memory

3.1.5.5 Reorganizing data interfaces

General

A data interface can be reorganized without having to interrupt or impair the RUN operating state.

Reorganizing the data interface

Certain central coupling blocks have a CDV connection (data type BOOL). At a positive edge at the CDV connection, the central coupling block inhibits coupling and formats the data interface on expiration of approx. 10 seconds. The data interface is then enabled again.

All transmitters/receivers go into a wait state during the inhibit time and reorganization. After the interface has been re-enabled, the channels will be registered and synchronized as during system startup.

This information is ignored for the local data interfaces of a CPU.

3.2 Local couplings in the rack

Overview

These local rack couplings include:

- Local CPU coupling
- Direct CPU-CPU coupling
- Buffer memory coupling

3.2.1 Local CPU coupling

General

A coupling module is not needed for local CPU coupling. This coupling type can only be used by function blocks that are located on the same CPU as the data interface. The data interface with a size of 1 MB is always available on the respective CPU.

Application

The coupling is primarily used to provide defined interfaces for self-contained tasks (e.g. a control system). This is a simple means of transferring the entire task to a different CPU if a CPU is "overloaded" in the course of configuration without involving extensive configuring work and then, for example, run communication via the data interface in buffer memory. For this purpose, you merely need to modify the CTS connection data on all communication function blocks.

Initialization and monitoring

The central blocks initialize and monitor the coupling in cyclic mode. @LOCAL coupling is therefore not enabled for all senders/receivers at the start of the cyclic mode and is delayed by several operating cycles. Once coupling has been enabled, the @LOCAL central block monitors the coupling.

Configuration

A @LOCAL central coupling block must be configured to initialize and monitor the coupling.

For local CPU coupling, you only need to specify the channel name at the AT, AR or US connections of the transmit/receive blocks. It is not necessary to specify any data for address levels 1 and 2. Transmitters and receivers having the same channel names can communicate.

3.2.2 Direct CPU-CPU coupling

General

The data interface for direct CPU-CPU coupling is provided locally on a CPU and has the following capacity:

- 1 MB (CPU550/551; VME)
- 1 MB (CPU555; VME)
- 4 MB (CPU555 and P0 coupling))

The data interface must be created on the CPU that contains the receive blocks, as the receive blocks cannot access the memory of the other CPUs via backplane bus. Memory access is only possible using the transmit blocks.

A configuration with the data interface on transmission side leads to a communication error ("C") at the receive block.

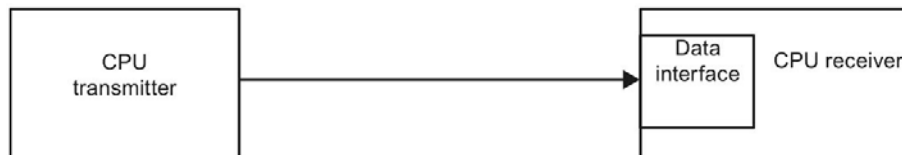


Figure 3-8 Direct CPU-CPU coupling

Application

Direct CPU-CPU coupling is used to exchange process data between different CPUs of a rack. In contrast to coupling with buffer memory, this method is more effective in terms of the transmission of larger data volumes. Using this coupling method, one communication partner (sender) accesses the data interface via the backplane bus, while the other (receiver) saves time by means of local access.

CPU555 enables an additional direct CPU-CPU coupling via the P0 bus to one/several other CPU555. The advantage of this coupling is even faster data transfer.

Requirement: The CPU555 modules must be located in slots with P0 connectors.

Initialization and monitoring

Configure a @LOCAL central block for initialization and monitoring of the coupling on the CPU on which a data interface is to be created.

Configuration

The @LOCAL central block initializes and monitors the coupling in cyclic mode.

This means that the coupling is not enabled for all transmit/receive blocks at the start of the cyclic mode, but is delayed by several operating cycles. Once coupling has been enabled, the @LOCAL central block monitors the coupling.

Direct CPU-CPU coupling can only be used by transmit/receive blocks that are configured in the same rack.

For direct CPU-CPU coupling, specify the channel name and also set an "E" as address level 1 at the AT or AR connections of the send/receive blocks. This data is optional for the CPU that must access via the backplane bus (sender).

It is not necessary to specify data for address level 2. Transmitters and receivers having the same channel names can communicate.

A ".P" must be specified as address level 1 via the P0 bus for direct CPU-CPU coupling.

Configuration information for direct CPU-CPU coupling.

Table 3- 3 Configuration information via backplane bus

Send end (CTV)		Receive end (RCV)	
CTS	Receive CPU	CTS	Local CPU
AT	Channel name.E	AR	Channel name.E

Table 3- 4 Configuration information via P0 bus (CPU555 only)

Send end (CTV)		Receive end (RCV)	
CTS	Receive CPU.P0	CTS	Local CPU
AT	Channel name.P	AR	Channel name.P

Note

Transfer mode

Only "Handshake" or "Refresh" transmission modes are possible.

3.2.3 Buffer memory coupling

General

The data interface for coupling memory coupling is located on a coupling memory with a size of 8 MB.

The hardware configuration is implemented using module CP50M1, CP51M1 or CP53M0.

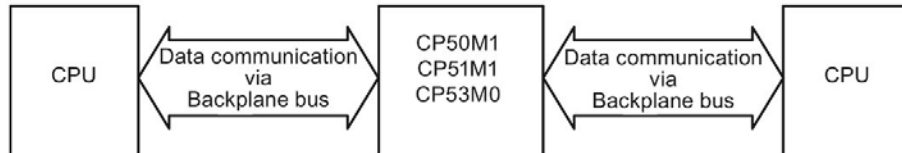


Figure 3-9 Buffer memory coupling

Application

Buffer memory coupling is used to transmit data between different CPUs of a rack. In contrast to \$ signals, this method is more efficient with regard to the transmission of higher data volumes.

Initialization and monitoring

Configure a @GLOB central block on any CPU of the rack for initialization and monitoring of the coupling.

Configuration

The @GLOB central block initializes and monitors the coupling in cyclic mode. This means that the coupling is not enabled for all transmit/receive blocks at the start of the cyclic mode and is delayed by several operating cycles. The @GLOB central block monitors the coupling once the coupling has been enabled.

Buffer memory coupling can only be used by transmit/receive blocks that are configured on the same rack.

You only need to specify the channel name at the AT-, AR- or US connections of the transmit/receive blocks for buffer memory coupling. It is not necessary to specify any data for address levels 1 and 2. Transmitters and receivers having the same channel names can communicate.

Note

Transfer mode

"Handshake", "Refresh", "Select" and "Multiple" are possible as transmission mode.

3.2.4 Application notes

Number of channels

The maximum number of channels through send and receive blocks (e.g. CTV and CRV) depends on the size and number of items of user data and on the access mechanism (Handshake, Refresh).

Calculation

The number of channels can be roughly calculated according to the following equation:

- **Per Refresh/Multiple channel:** 150 bytes + 2 x number of user data bytes (size of the virtual connection)
- **Per Handshake/Select channel:** 150 bytes + 1 x amount of user data bytes

3.3 CP52M0 rack coupling

3.3.1 Areas of application

GlobalDataMemory (GDM) facilitates data exchange between all racks and CPU modules in the system.

A separate rack is used for GDM (as of herewith referred to as **GDM rack**). The **CP52M0 memory module (slot 1)** and a corresponding number of **CP52IO interface modules (slots 2-12)** are inserted in the rack that provides 21 slots.

Each rack that is coupled with GDM (hereinafter referred to as **coupled racks**) must contain a **CP52A0 access module**. These racks are interconnected with GDM by means of fiber-optic cables in a star topology.

Note

Removal and insertion of the fiber-optic cables is not permitted during operation.

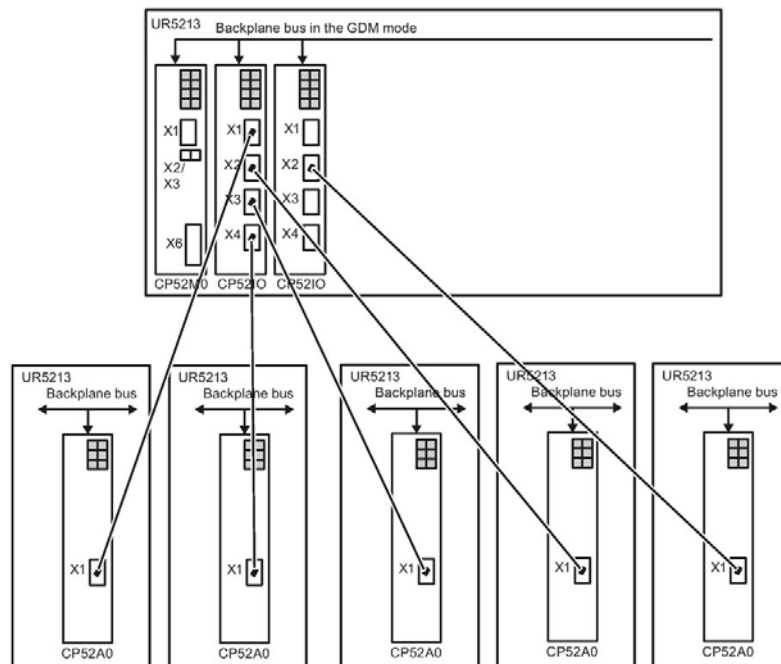


Figure 3-10 Example of a GDM system with 5 racks that are coupled with FOCs

CP52M0 function

The CP52M0 GDB memory module provides 2 MB of central memory for the GlobalDataMemory system. This memory module handles the entire data traffic between the processors in the coupled racks.

Data is exchanged between the GDM interface modules CP52IO and the CP52M0 via the backplane bus.

References:

System manual "SIMATIC TDC hardware"

3.3.2 Power on/off response

- You can choose any sequence to switch on the racks.
- All frame channels are shut down in the coupled racks if the GDM rack that is switched off at runtime. These frame channels are re-initialized after the GDM rack has been switched on again.

Note

If you switch on the GDM rack while the coupled racks are in operation, you must expect an increased computing time when activating the communication link for CPUs that communicate via rack coupling. This scenario may lead to the display of an 'E' (error in Task Manager) on CPUs that are already subject to heavy load.

- Power to the coupled racks can be cycled off and on at runtime. However, this state will prevent further data exchange between the coupling partners and this rack.
- Communication between the remaining nodes (GDM rack and a maximum of 43 coupled racks, each with a CP52A0) are not affected if power to one of the coupled racks is cycled off and on.
- Coupled racks that are switched off can be reconfigured and switched on again. You may also change number of transmitters and receivers (e.g. the configuration is lacking one transmitter).

Note

Modification of the data length with existing coupling leads to communication errors in the rack in which the configuration was changed. In this case, you also need to reset the GDM rack.

- As soon as the rack is switched on again, a new connection is established between the GDM rack and the link partner that was switched on again. The frame channels are initialized and communication is resumed without having to reset the link partners.

3.3.3 Synchronization and trigger options

Access module CP52A0 provides the following synchronization and trigger signals:

1. Basic clock cycle
2. Bus interrupt
3. Time

The clock time and basic clock cycle synchronization signals can be transmitted separately only by one CP52A0. CP52A0 supports transmission and receiving of these signals. The decision to transmit or receive (basic clock cycle), or only to transmit (clock time), is made in HWConfig.

If the clock time and the basic clock cycle was configured for transmission on several access modules, the signal is always transmitted from the CP52A0 that is interconnected with the CP52IO whose FOC interface is located on the extreme left of the rack.

The SYSFAIL* and bus interrupt signals can transmitted and received simultaneously on every CP52A0.

3.3.4 Configuration

Rules

1. The entire user configuration is completed in the coupled racks.
2. The coupled racks (UR5213) must be assigned **unique** names ("General" tab in the object properties dialog).
3. A CP52A0 GDM access module must be configured as communication module for each coupled rack in a user-defined slot in HWConfig.
4. Only **one** CP52A0 GDM access module may be configured per coupled rack.
5. A @SRACK central block must be configured for each CP52A0.

3.3.4.1 Configuration in HW Config

The hardware configuration of a rack is defined in HWConfig. The CP52A0 module is available in the SIMATIC TDC module catalog, communication modules.

Select the "Synchronization" tab to specify whether to transmit, receive, or deactivate the basic clock cycle, bus interrupt, and time signals:

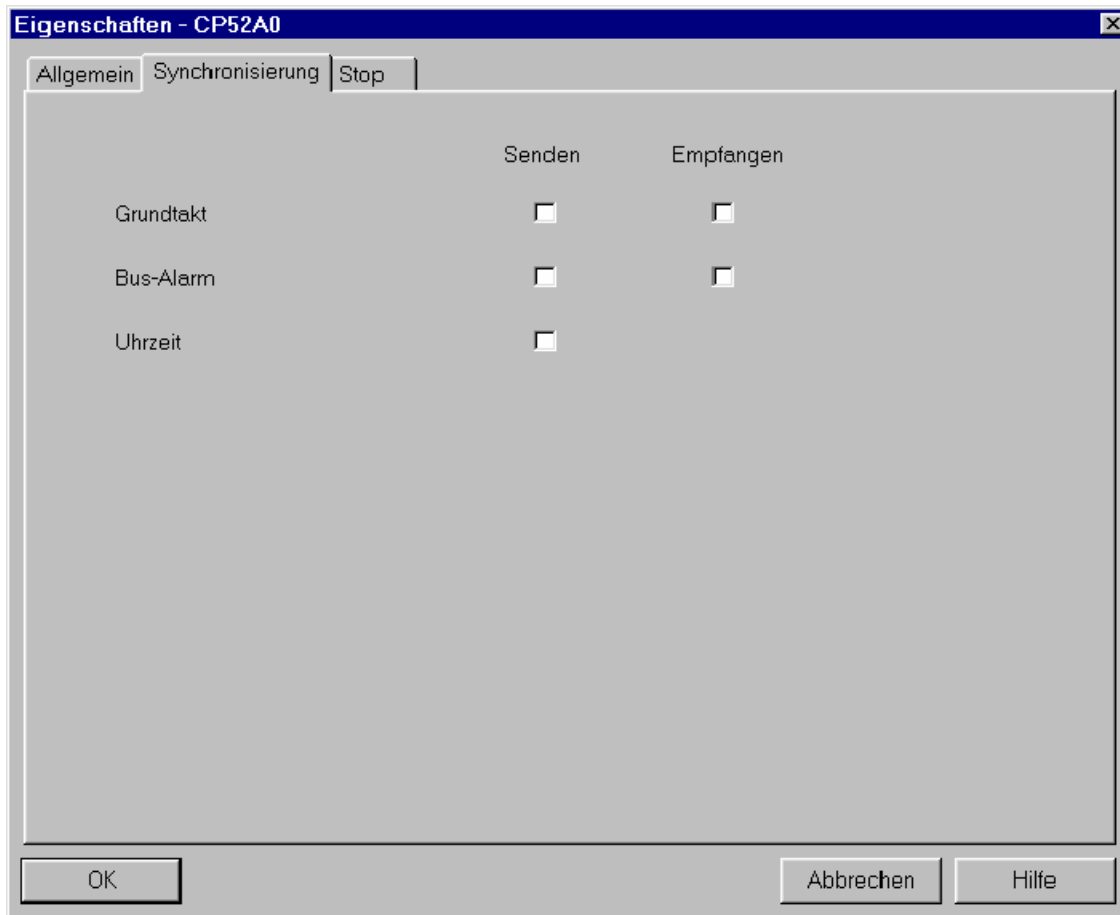


Figure 3-11 "Synchronization" tab

The following states are available for the **basic clock cycle** signal:

1. Receive signal, **or**
2. Transmit signal, **or**
3. Deactivated

The following states are available for the **bus interrupt** signal:

1. Receive signal, **and / or**
2. Transmit signal, **or**
3. Deactivated

The following states are available for the **time** signal:

1. Transmit signal, **or**
2. Deactivated

Note

The reception of the time signal is set at the RTCM block.

The signals are deactivated if none of the buttons are selected.

Default setting

The signals are in **deactivated** state (nothing selected) by **default**. You can click the button to specify whether to transmit or receive a specific signal, or retain the basic setting.

In the "Stop" tab, you can specify whether or not the module stops the entire rack on receiving a system error signal (SYSFAIL*) from other racks. At the same time, you can specify whether or not to transmit this signal.

Note

As long as a coupled rack displays a flashing "H" (SYSFAIL active) and transmits the SYSFAIL signal due to the configuration, the other coupled racks will also be prevented from starting. In this case, switch off the master rack and reset the coupled racks that cause the SYSFAIL signal.

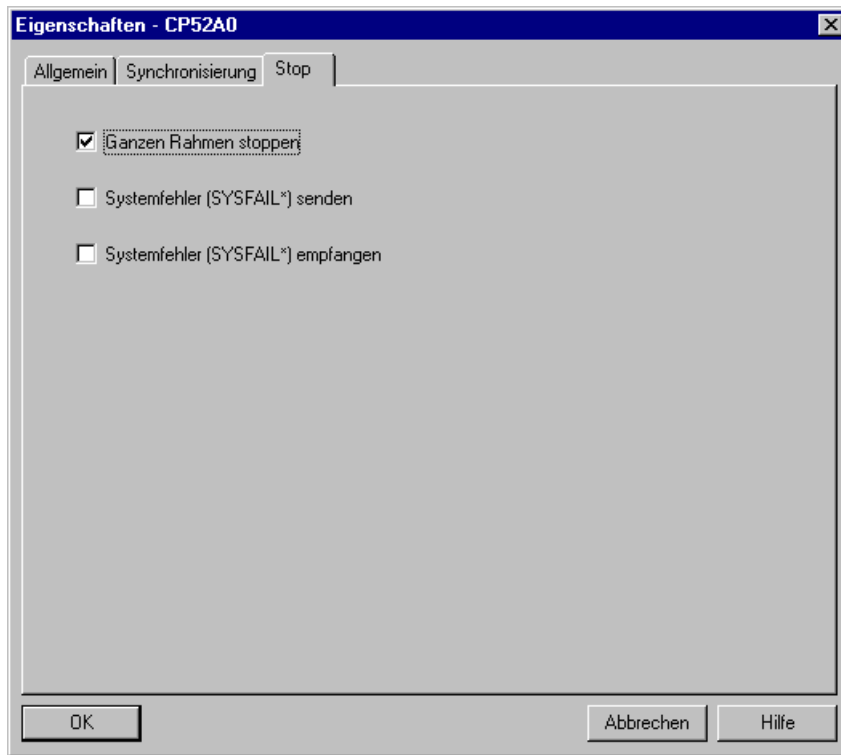


Figure 3-12 "Stop" tab

3.3.4.2 Configuration in CFC

Configure the @SRACK central block for the CP52A0 module of the coupled rack.

You always have to configure the "CTS" interface (CP52A0 module name).

The rack to be monitored can be specified at the "Nxx" connections (name of rack xx) starting from N01 (and continued contiguously in ascending order with N02, etc.). The block evaluates the connection assignment until the first connection with an empty string.

Data exchange is configured by means of transmit and receive blocks. For rack coupling, you only need to specify the channel name at the AT and AR inputs. It is not necessary to specify any data for address levels 1 and 2. Transmitters and receivers having the same channel names can communicate.

3.3.5 Performance data

Transmission rates

The data transmission rate between the racks is dependent on the configuration to a greater extent.

For a configuration of approx. 20 racks, each with 4 CPUs, you can assume a data throughput of **60 bytes/ms** per CPU for transmission and receiving.

Cable lengths

The performance of the rack coupling depends on the length of the fiber-optic cables used. It is therefore recommend to use the shortest possible cables lengths. Longer cables slow down memory access and, increase the computing load on the CPU, and reduce the maximum possible data transmission rate.

The maximum cable length for a connection between CP52A0 and CP52IO is 200 m.

Interface assignment

The last assigned interface for each CP52IO provides a slightly wider time window for GDM access. At the same data throughput, this reduces computing time load on the CPU551.

Example

Seven racks are connected to the GDM rack via two CP52IO.

Interfaces X1 to **X4** are used on the first CP52IO, and interfaces X1 to **X3** on the second.

This means that the **X4** (first CP52IO) and **X3** (second CP52IO) interfaces generate less CPU computing load on the connected racks.

3.4 CP53M0 rack coupling

General

The following couplings can be realized using the CP53M0 module:

A SIMATIC TDC rack can be connected to a SIMADYN D system with:

- SIMADYN D as master:

A CP53M0 can be connected instead of a CS22 to the CS12/CS13/CS14 at any position. The CP53M0 should be parameterized for operation in slave mode.

- SIMATIC TDC as master:

A CP53M0 is parameterized for operation in master mode in a SIMATIC TDC rack. A CS22 or a CP53M0 operating in slave mode can be connected to the two FOC interfaces.

In addition to the coupling to SIMADYN D, you can link up to three SIMATIC TDC racks. The CP53M0 is parameterized accordingly for operation in master mode in one rack and in slave mode in the other two racks.

The rack containing the CP53M0 module for operation in master mode is referred to in the following as master rack.

The rack containing the CP53M0 module for operation in slave mode is referred to in the following as slave rack.

The coupling via CP53M0 in master or slave mode is configured similar to a coupling on SIMADYN D side.

Initialization and monitoring

One @CS1 central block (master mode) or @CS2 (slave mode) must be configured on any CPU in each rack for coupling initialization and monitoring.

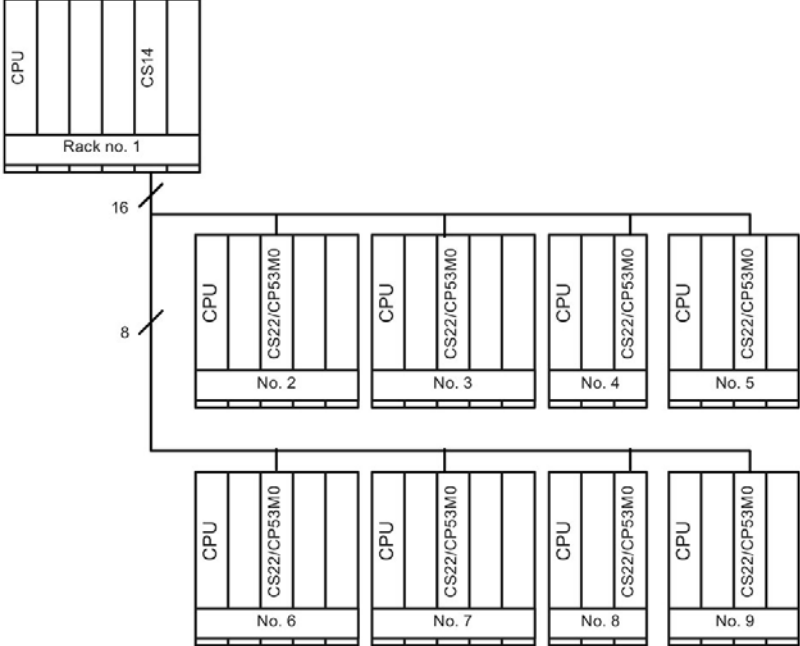


Figure 3-13 Maximum configuration for 8 Slaves with CS14

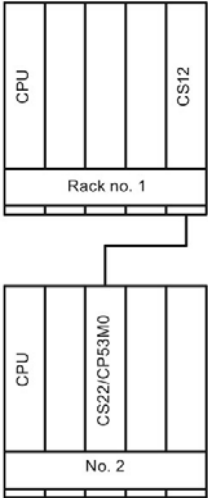


Figure 3-14 Point-to-point connection with CS12

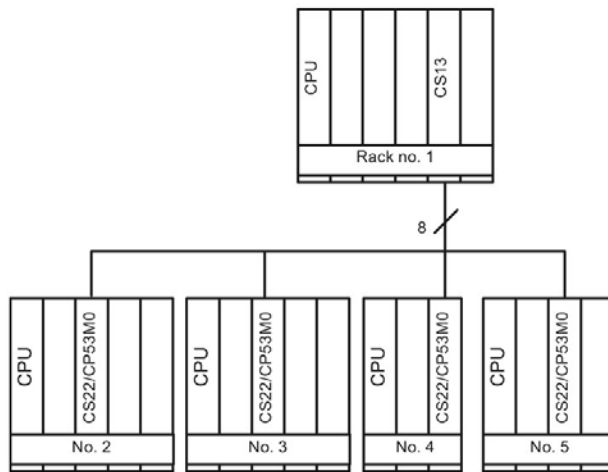


Figure 3-15 Configuration for four slaves with CS13

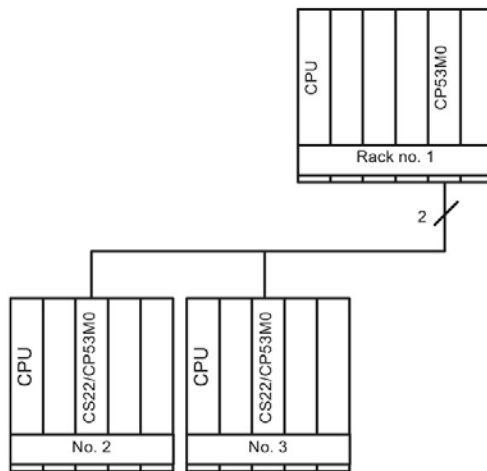


Figure 3-16 Maximum configuration with CP53M0 as master

3.4.1 Hardware installation

Overview

- Only SIMADYN D racks with C-bus connection can be coupled to CP53M0 (e.g. SR24).
- The master rack contains one or several CS12, CS13, CS14, or CP53M0 modules, depending on the number of slaves to be connected. The slave racks contain a CS22 or CP53M0 module.
- You can couple up to two slave modules (CS22 or CP53M0 in slave mode) to a CP53M0 operating in master mode.
- Several CP53M0 can be configured for operation in slave or master mode in a SIMATIC TDC rack. This means that it is possible to configure several different rack couplings in a rack.
- The CP53M0 modules of a rack coupling must be configured in different racks.

3.4.2 Scope of performance

Overview

All slave racks are permanently coupled to the master subrack, as a slave rack must continuously access the master rack memory.

- The master/slave racks can be switched on in any sequence.
- Power to the racks can be cycled off and on at runtime.
- Cycling power to a slave rack off and on does not affect communication between the remaining nodes.
- You can switch off the slave racks, reconfigure these and then switch on power again. You may also change number of transmitters and receivers (e.g. the configuration is lacking one transmitter).
- As soon as the slave partner is switched on again, a new connection is activated between this slave partner and all other partners. This is also valid for slave-slave communications, i.e. if the CS12-, CS13-, CS14 or CP53M0 module (master mode) is only used as data exchange area and not as communication partner. However, slave-slave communication will be interrupted if the master rack is switched off.

Note

Removal and insertion of the fiber-optic cables is not permitted during operation.

3.4.3 Response to the "shutdown" of a coupling partner

Response of the master rack

The master rack will be switched off.

The @CS2 central block can no longer access the master rack and prepares a restart (in addition, the CDM block output is set to the "error" state, refer to @CS2 mask). The system then goes into wait state until the master rack has been switched on again.

The slave transmit/receive blocks are no longer able to access the master rack and initiate a new channel registration.

Response of the slave rack

The slave rack will be switched off:

The @CS1 central block and the maximum seven additional @CS2 central blocks decrement their particular NCP connection (i.e. the number of active slave racks is reduced by the count of one). Otherwise, there is no response and the NCP connections are incremented after the restart of the corresponding slave rack.

All configured transmit/receive blocks whose coupling partner is located on the rack that is switched off go into wait state until the rack is restarted.

3.4.4 Response to "power on" of the master rack

Reaction

If the master rack is switched on again while the slave racks are in operation, you must expect a short-term increase of computing time requirements when activating the connection for CPUs that communicate via rack coupling. This scenario may lead to the display of an 'E' (error in Task Manager) on CPUs that are already subject to heavy load.

Acknowledgment

The 'E' can be acknowledged in two ways:

Manual acknowledgement

Once the connection has been restored, you can acknowledge the 'E' display on the respective CPU by pressing the acknowledge button.

Automatic acknowledgment

To enable automatic acknowledgment, the following settings must be configured on all CPUs in the slave rack that communicate via rack coupling. Automatic acknowledgment can be realized in two different ways based on this configuration:

1. All YEV outputs of the FBs that communicate by means of rack coupling are monitored a configured function (see Fig. 3-3). Input NOT.I is set to '1' if the value of all YEV outputs is less than 9 (i.e. initialization has been completed). With the help of output CDM of the @CS2 central block it is ensured that automatic acknowledgment is only set if the master rack is actually switched on. A timeout (input T at PCL) specifies the timeout for automatic acknowledgment. The 'E' output to the seven-segment display is now automatically acknowledged using the SYF4 function block.

2. If it is not possible, or if you do not want to monitor all YEV outputs, set input OR.I2 to "1" and do not interconnect input NOT.I. In this case, the CPU is only acknowledged within the timeout et at connection T of the PCL after the master rack has is switched on (output of @CS2.CDM).

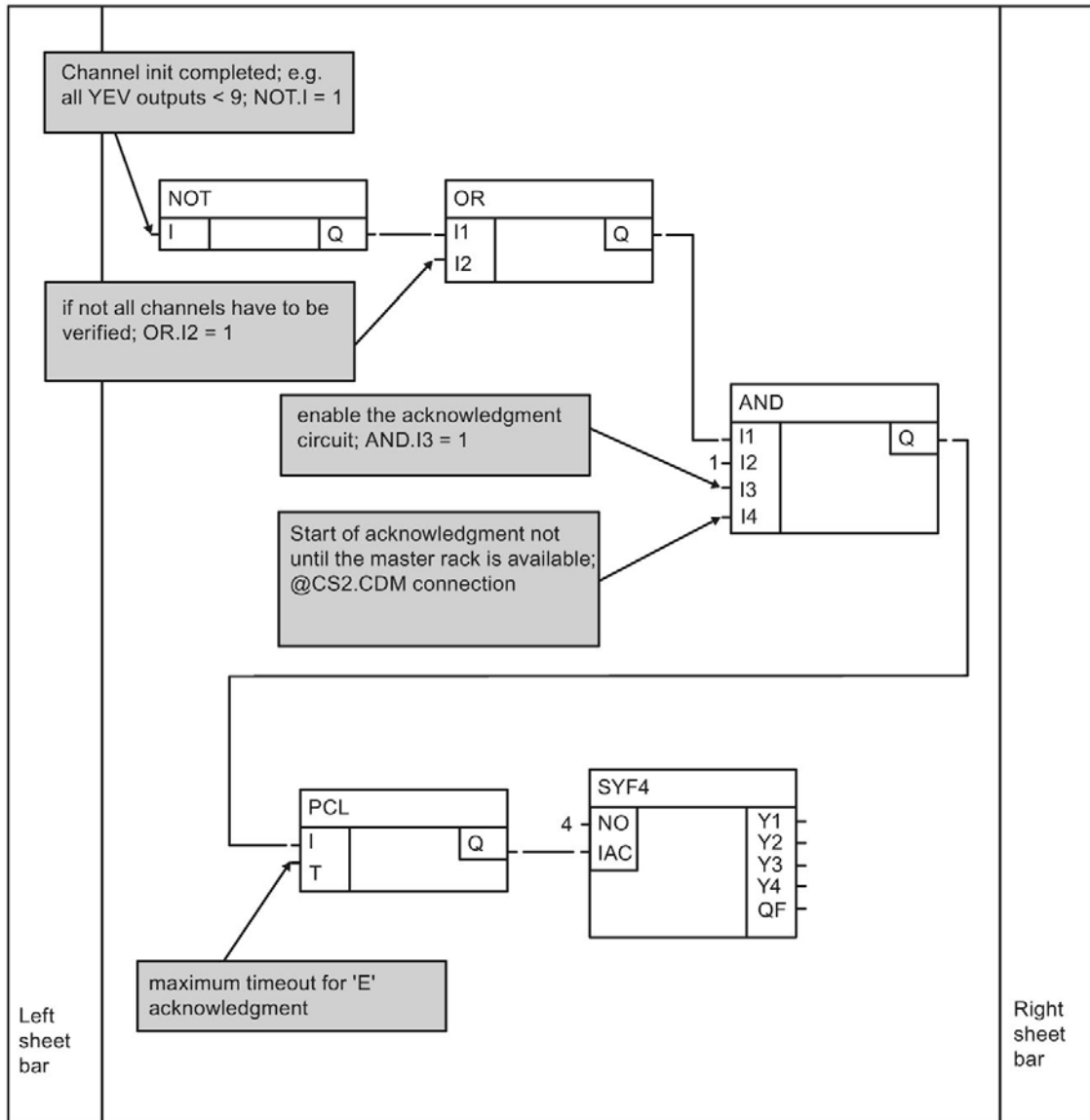


Figure 3-17 Automatic acknowledgment of 'E'

3.4.5 Restart capability

Synchronizing transmitters and receivers

The restart capability of transmitters/receivers is a further important communication feature for external communication interfaces. Transmitters/receivers will always find and synchronize their old channel.

You can cycle power to the racks off and on in any order. The transmitters/receivers of the racks containing a CP53M0 (slave mode) are automatically synchronized with the old channel at each restart.

A transmitter/receiver identifying a "suitable" channel at logon will be unable to identify

- whether or not it has "previously" used this channel, OR
- whether OR NOT this channel is currently in use by another transmitter/receiver.

3.4.6 Configuration

Rules

- All CP53M0 modules (slave mode) must be assigned different names for rack coupling by means of fiber-optic cables rack coupling. If redundant names are found, the corresponding central blocks with multiple configuration (FB disable) will log off.
- All CP53M0 modules of a rack coupling must be inserted in different racks.
- The sampling time range of $32 \text{ ms} \leq T_A \leq 256 \text{ ms}$ is valid for central coupling blocks and the central blocks @CS1 and @CS2 for rack coupling. Note that the sampling time configured at the @CS2 central blocks must be a multiple of the sampling time set at central block @CS1 (the sampling times may be identical for basic clock synchronization).

The actual sampling time (in milliseconds) is decisive instead of the cyclic task (T1, T2, etc.)

Data interface

The data interface is available in dual-port RAM of the CP53M0 module (master mode). The data exchange area has a length of 128 KB.

Initialization and monitoring

The @CS1 and @CS2 central blocks initialize and monitor the coupling in RUN operating state. This means that coupling is not enabled for all transmit/receive blocks at the start of the cyclic mode and is delayed by several sampling cycles. You always have to activate the coupling in the master rack before you activate it in the slave racks.

The @CS1 and @CS2 central blocks monitor the coupling once it has been activated. The central blocks output the number of active coupling partners accordingly.

Data specified at the AT and AR inputs

You merely need to specify the channel name at the AT and AR inputs of the transmit/receive blocks for rack coupling. It is not necessary to specify any data for address levels 1 and 2. Transmitters and receivers having the same channel names can communicate.

3.4.7 Restrictions

Observe the following specifications for data exchange with a SIMADYN D rack that is configured with STRUC:

- Only the CTV / CRV may be used for data exchange in a SIMATIC TDC configuration. The use of CTV_P / CRV_P blocks (pointer blocks) leads to a communication error ('C').
- Only the data types I2, I4, and Real (NF) can be coupled in STRUC. All other data types lead to a communication error ('C'). However, in this case it is possible to use the type switching function in STRUC.

Observe the following timing difference between master and slave access via backplane to the CP53M0 module:

- Master: approx. 1 μ s (4 bytes)
- Slave: approx. 8 μ s (4 bytes)

With the same data volume, load on the CPUs in the slave rack increases accordingly.

3.5 TCP/IP coupling (CPU555; CP51M1)

Introduction

This configuration model describes the basic procedures configuration engineers need to follow when implementing the coupling of a SIMATIC TDC rack via TCP/IP or UDP.

This configuration model defines the basic hardware equipment and the function blocks and demonstrates the principles for using these. The functional scope of this configuration model was intentionally kept to a minimum in order to get you started quickly. You may always expand functionality and/or the hardware components. However, you must always observe the specifications in the respective function block documentation.

All designations used in this configuration are selected at randomly and are only binding for this model.

The structure of these configuration instructions reflects the chronological order of steps to take for creating the entire configuration. However, these instructions only represent a recommendation and are not mandatory.

Applications

CP51M1 or the PROFINET interface of the CPU555 can be used for the following applications:

- Exchange process data with
 - Another CP51M1
 - PROFINET interface of the CPU555
 - SIMATIC Industrial Ethernet module (e.g. CP443-1)
- Visualization of process data in WinCC
- Visualization of messages in WinCC
- Exchange of process data with third-party systems (e.g. process computers)
- Central commissioning and diagnostics of all CPU modules in the rack
- Time synchronization for uniform system time
- Routing is supported in FW V1.1 or higher and in D7-SYS V7.0 or higher.

Note

Request the IP, subnet and router addresses, as well as the port numbers for applications from your network administrator when integrating new modules into an existing TCP/IP network.

References:

- TCP/IP basics
(e.g. W. Richard Stevens, TCP/IP Illustrated, Volume 1, Addison-Wesley Publishing)
- System manual SIMATIC TDC hardware
(<http://support.automation.siemens.com/WW/view/de/8776697/0/en>).

3.5.1 Comparison of TCP/IP with UDP

TCP/IP

The TCP/IP protocol is connection-oriented, which means that data can only be transmitted if the coupling partner is actually available. As long as there is a connection to the coupling partner, the TCP/IP protocol ensures that the data transmitted will be received by the coupling partner. In the case of error, several retries are made to transmit the data.

An unidirectional connection shutdown may lead to protocol-related **data loss** in the TCP/IP stack of CP51M1 or at the coupling partner (e.g. PC).

Note

For important information, data reception must be monitored at **application level**.

UDP

The UDP protocol is not connection-oriented, i.e. data is also transmitted even if the coupling partner is temporarily unavailable, which means that **the data is lost inevitably**. However, received data is correct as a result of the data backup mechanisms.

In comparison with TCP/IP, UDP achieves a higher communication speed between the coupling partners. However, this higher rate is only possible at the cost of restricted data transmission reliability.

Channel modes

The TCP/IP and UDP protocols support the following channel modes:

- Handshake
- Refresh
- Multiple
- Select

Typical applications

The **Refresh** and **Multiple** channel modes are overwriting modes (i.e. older data can be overwritten by more recent data). The **UDP** data transmission protocol is therefore particularly suitable for these two channel modes.

Data is not overwritten in the **Handshake** and **Select** channel modes, which is why **TCP/IP** is recommended as data transfer protocol.

Client or server

The following table provides an overview of the conditions for operating SIMATIC TDC as TCP/IP client or server:

SIMATIC TDC (TCP/IP) process data (CRV/CTV)		Communication partner
Specification of the IP address and port at address level 2	Connect → ← accept Send data →	Must be server
Not specified in address level 2	← Connect accept → ← Send data	Must be client

3.5.2 Configuration model

You need at least the following hardware components per station to implement coupling over TCP/IP for SIMATIC TDC:

- Rack UR6021
- CPU module CPU555 with memory module
- CP51M1 communication module

3.5.3 Configuration steps

The following paragraphs demonstrate only the configuration steps for a single station. All other stations are handled likewise.

3.5.3.1 Configuration in HW Config

Parameterizing CP51M1

CP51M1 is parameterized in the corresponding tab of the object properties dialog.

To parameterize the Ethernet interface, the following relevant settings must be specified in the "Properties of the CP51M1 Ethernet interface" sub-tab:

- **IP address ("Parameters" tab)**
IP address of the module in dot notation. In this case: **141.20.135.197**
- **Subnet mask ("Parameters" tab)**
Subnet mask for identification of the network segment
In this case: **255.255.0.0** a class B network
- **IP address of the default router ("Parameters" tab)**
Here: **"Do not use router"**

Note

Verify that the IP address and subnet mask are harmonized. For example, class B subnet mask 255.255.0.0 is invalid for a class C IP address (193.x.y.z). For more information on the selection of an IP address and its range of values, refer to the CP51M1 Online Help.

3.5.3.2 Configuration with CFC

The following paragraphs cover only the most important I/Os. Deviations resulting from the configuration of a UDP coupling are described in the corresponding sections.

Required FBs

The following function blocks are required for this sample configuration:

- **@TCPIP** central block, for initializing and monitoring the CP51M1 module (**always required**)
- **CRV** receive block (in this case, optional for process data coupling)
- **CTV** transmit block (in this case, optional for process data coupling)

Note

CPU555

The function block @PNIO should be used for all PROFINET, TCP/IP and UDP communication on the CPU555.

Central block @TCPIP

I/O	Pin assignment (meaning)
CTS	D1800C.X01 (module name and connector identifier for the configured CP51M1)

Receive block CRV - "AR" I/O

Initialization connection for address data. In addition to the channel name, you must also specify **address level 1** and address level 2 (only for clients).

The two address levels are separated in the notation by a dot ".".

Rules for address level 1:

- The first character (letter) specifies the protocol ("**T**" = TCP/IP, "**U**" = UDP) (in this case, "**T**").
- The second character must be a hyphen "-".
- The next **5 digits** specify the channel port number, whereby leading zeros must be specified (in this case: "**01024**" for port number 1024).

Note

Only the port numbers 1024 to 65535 should be used in agreement with the system administrator. Port numbers up to and including 1023 are usually reserved for "Well-known services" and "Unix-specific services".

Rules for address level 2:

- The first **12 digits** specify the IP address of the remote coupling partner. This is entered in the so-called "dot notation", but without separating period. Leading zeros must be specified (here: "**141020135198**" for IP address 141.20.135.198).
- The 13th character must be a hyphen "-".
- The next **5 digits** specify the port number of the remote coupling partner, whereby leading zeros must be specified (in this case: "**01024**" for port number 1024).

Configuration

I/O	Pin assignment (meaning)
CTS	D1800C.X01 (module name and connector identifier for the configured CP51M1)
AR	RXKAN1.T-01024 (receive address parameter)
MOD	H (receive mode)
EN	1 (enable)

Transmit block CTV - "AT" I/O

Initialization connection for address data. In addition to the channel name, you must also specify **address level 1** and **address level 2** (only for clients).

The two address levels are separated in the notation by a dot ".".

Rules for address level 1:

- The first character (letter) specifies the protocol ("**T**" = TCP/IP, "**U**" = UDP) (in this case, "**T**").
- The second character must be a hyphen "-".
- The next **5 digits** specify the channel port number, whereby leading zeros must be specified (in this case: "**01024**" for port number 1024).

Note

Only the port numbers 1024 to 65535 should be used in agreement with the system administrator. Port numbers up to and including 1023 are reserved for "Well-known services" and "Unix-specific services".

Rules for address level 2:

- The first **12 digits** specify the IP address of the remote coupling partner. This is entered in the so-called "dot notation", but without separating period. Leading zeros must be specified (here: "**141020135198**" for IP address 141.20.135.198).
- The 13th character must be a hyphen "-".
- The next **5 digits** specify the port number of the remote coupling partner, whereby leading zeros must be specified (in this case: "**01024**" for port number 1024).

Configuration

I/O	Pin assignment (meaning)
CTS	D1800C.X01 (module name and connector identifier for the configured CP51M1)
AT	TXKAN1.T-01024.141020135198-01024 (transmit address parameter)
MOD	H (transmit mode)
EN	1 (enable)

3.5.4 Application notes

Number of channels

A maximum of **128 channels** can be set up on the TCP/IP module using transmit and receive blocks (e.g. CTV and CRV).

The actually possible number of channels depends on the size and number of user data and on the access mechanism (Handshake, Refresh).

The following are available for the data interface:

- CPU555: 384 KB RAM
- CP51M1: 254 KB RAM

Calculation

The number of channels can be roughly calculated according to the following equation:

- **Per Refresh/Multiple channel:** 150 bytes + 2 x number of user data bytes (size of the virtual connection)
- **Per Handshake/Select channel:** 150 bytes + 1 x amount of user data bytes

Frame length

The frame length is restricted as follows:

Protocol	User data length for transmit and receive mode, respectively
TCP	32767 bytes
UDP	1472 bytes

"Ping" on CP51M1

Communication with CP51M1 can be checked using a "Ping". The module only responds to a "Ping" if the communication partner is located in the same network or a default router (within the same network) is able to activate the connection.

Performance

The performance of TCP/IP coupling depends on the configuration used. The following data indicate the performance values for a typical configuration (do not necessarily represent the maximum values).

Requirements

- For frame lengths 192 and 1024 bytes, 32 transmit and 32 receive UDP connections are configured for each CP51M1 (Refresh).
- For a frame length of 4096 bytes, 8 transmit and 8 receive connections are configured.

Results

Approx. **1270** transmission/receive tasks are available for each CP51M1

The maximum data rate is approx. 1 MB/s (frame length of 1024 bytes); this cannot be increased, in spite of the higher channel lengths.

Data throughput Kbps	Tasks/s	Transmit/receive cycle ms / connections	Frame length bytes
302.260	1270	64 / 40	192
974.556	911	64 / 56	1024
888.16	208	64 / 16	4096

3.5.5 Communication via WinCC

The communication to WinCC is performed via a standard coupling (without additional software in WinCC; only access to process data possible).

Standard coupling

For the configuration at the TDC end, proceed as described in section WinCC connection to SIMATIC TDC via standard channel (SIMATIC S7 Protocol Suite.CHN) (Page 239).

Note

Access to system connections is only allowed using data block addressing. Access using bit memories is not allowed and leads to an error message in WinCC!

3.5.6 Central service

Section Communication utility service (Page 232) describes the procedures for enabling central services (e.g. CFC Online) via CP51M1 / CPU555.

3.5.7 Time synchronization

For more information on time synchronization, refer to chapter Communication utility time synchronization (Page 236).

3.5.8 Migration from CP5100 to CP51M1

To migrate a configuration with CP5100 to the new CP51M1 module, proceed as follows:

- Delete CP5100 in HW Config and configure CP51M1 (verify that the same data is specified as for CP5100, e.g. IP address).
- In the CFCs, replace the CP5100 name at all CTS I/Os of CTV/CRV blocks with the CP51M1 module name and connector identifier (e.g. D1800C with D1800C.X01). Corresponding support is available to you by opening a chart for each CPU and selecting Options --> Convert CTS I/O. You only need to specify the two names (name that exists at the CTS I/Os, plus the new name to be entered) in the next dialog. If the names are correctly entered and can be replaced, they will be replaced in the chart container. In the case of error, a corresponding error message is output.
- Manual intervention necessary if UDP frames with a length > 2048 bytes were configured. CP51M1 only supports UDP transmission of 2048 bytes!

3.6 PROFIBUS DP coupling (CP50M1)

3.6.1 General basics

Properties

The CP50M1 has the following properties on PROFIBUS DP:

- **Master/Slave**

Each of the two interfaces of CP50M1 can be operated on PROFIBUS DP in master mode (standalone, or with other masters in multi-master mode) and in slave mode. This can be implemented separately for each interface.

- **Shared input**

All slaves connected to PROFIBUS DP are assigned exactly one master (the parameterizing master) and are initially only able to communicate with this master. Additional masters can read the slave input data using the "Shared input". The interfaces of CP50M1 support this functionality as master.

- **Routing**

CP50M1 FW V1.1 or higher and with D7-SYS V7.0 or higher support the routing function.

- **SYNC and FREEZE**

The SYNC and FREEZE utilities enable synchronous read/write access to the I/O of multiple slaves. CP50M1 supports these utilities in master mode.

- **Constant bus cycle time**

This property of PROFIBUS DP ensures bus cycles of exactly the same length.

- **Direct data exchange**

The configured slaves can "directly" exchange data without configuration in CP50M1.

- **Data lengths**

A bidirectional transmission of up to 244 bytes per slave is supported.

- **Consistency**

The data within a frame is always consistent.

Note

Up to six CP50M1 can be operated simultaneously in the same rack.

3.6.2 Configuration

Configuring the DP system on CP50M1

The DP system for CP50M1 is configured as in SIMATIC with HW Config and in networking. This means that it is configured exactly the same as other DP masters (e.g. CPU 315-2DP).

Details of the procedure are available in the "Configuring hardware and connections in STEP 7" Manual, section 3 "Configuring distributed peripherals (DP)" and section 11 "Networking Stations".

For this reason, the following paragraphs only cover the special features of CP50M1.

Configuring communication in CFC

The following function blocks must be configured for PROFIBUS DP coupling:

- A central coupling block @PRODP
- A maximum of one transmit and receive function block per slave station
- One synchronizing function block SYNPRO can be configured
- One DPDIAG diagnostics function block and one diagnostics function block per slave can be configured

Communication service

The following communication utilities are allowed:

- Process data
- Parameter processing of variable speed drives

Transfer mode

Valid transmission mode:

- Refresh

For receivers, optionally also multiple

Central coupling block

The @PRODP central coupling block initializes and monitors the PROFIBUS DP coupling via connectors X1 und X2 of CP50M1.

Data specified at address connection AT, AR

Special features of the data specified at the address connection of the AT, AR transmit/receive blocks when using PROFIBUS DP:

Input sequence:

"Channel name.Address level 1.Address level 2"

- **Channel name:**
 - Max. 8 characters
 - ASCII characters except "dot" and @
 - Channel names of all transmit and receive blocks that access the interfaces X1 and X2 of CP50M1 must be different (exception for the "Multiple" transmission mode).
 - The channel name has no particular significance for PROFIBUS DP.
- Append a dot "." to the channel name entry
- **Address level 1:**
 - The slave PROFIBUS address is specified as address level 1.
 - The slave PROFIBUS address must not be assigned more than once for each transmit and receive channel.
 - Range of values: 0, 3 - 123
 - 3...123: Addressing external slaves
- Append a dot "." to the address level 1 entry
- **Address level 2:**
 - Consists of up to two characters, of which the second is of no significance to CP50M1.
 - **1st character:** Byte ordering

"1": PROFIBUS default

The data is transmitted in "Motorola format" (most significant before least significant byte)

"0": Exception setting

The data is transmitted in "Intel format" (least significant before most significant byte)
This setting can be used for communication partners whose internal data management is based on the Intel format (e. g. SIMATIC TDC).

Examples of data specified at the address connection

- AT 'Setpoint.25.1'
 - The channel named **setpoint** transmits data to a **slave** at PROFIBUS address **25**.
- AR 'RECEIVE.117.0'
 - The channel named **RECEIVE** receives data from a **slave** at PROFIBUS address **117**.
As an exception, data is transmitted in **Intel format**.

Configuration as slave

The procedure for configuration as a slave is described in detail using an example in "D7-SYS - Example project SIMATIC TDC CP50M1 as PROFIBUS DP slave". You can find the document from the Windows taskbar "Start > Siemens Automation > SIMATIC > Documentation".

Shared input

The procedure for configuration is described in detail using an example in "D7-SYS - Example project SIMATIC TDC CP50M1, Direct Data Exchange DX". You can find the document from the Windows taskbar "Start > Siemens Automation > SIMATIC > Documentation".

Notes on configuration

With CP50M1, data can only be read from slaves that support the "direct data exchange" function. Suitable slaves are listed in the hardware configuration.

3.6.3 Constant bus cycle time

Introduction

A constant bus cycle time on PROFIBUS DP is configured for CP50M1 as for a SIMATIC CPU (also refer to the Manual "Configuring Hardware and Communication Connections STEP 7" Manual, chapter 3.12 "Setting Constant Bus Cycle Times for PROFIBUS Subnets").

3.6.4 SYNC/FREEZE commands

General

The SYNC and FREEZE commands synchronize the inputs and outputs of a group of slaves. The SYNPRO function block triggers these commands and supports consistency assurance.

Consistency

The configuration engineer is responsible for data consistency. The SYNC/FREEZE command is focused on data consistency at all participating slaves. It goes without saying that the consistency of the input or output data of a slave is always guaranteed.

SYNC

After a SYNC command has been triggered, the DP master (CP50M1) goes into wait state for the duration of one DP bus cycle to ensure that all slaves can receive the new output values and then transmits a SYNC broadcast frame to the configured slave group. All slaves of this group will then simultaneously update their buffered outputs.

The outputs are only updated again in cyclic mode after the DP master has broadcast the UNSYNC control command (EN=0 at block SYNPRO).

Consistency assurance:

It must be ensured in the configuration that SIMATIC TDC CPUs are prevented from changing the data within the DP bus cycle time that follows a SYNC command trigger.

FREEZE

The DP master broadcasts a FREEZE frame to the configured slave group immediately after a FREEZE command was triggered. All slaves of this group will then simultaneously read and buffer their inputs. This input data will be available on the SIMATIC TDC CPUs on expiration of a DP bus cycle time.

The DP slave transmits the input data to the DP master again in cyclic mode after the DP master has broadcast the UNFREEZE control command (EN=0 at block SYNPRO).

Consistency assurance:

It must be ensured in the configuration that the DP master is prevented from evaluating the input data within the DP bus cycle time following the FREEZE command trigger.

3.6.4.1 SYNC/FREEZE configuration variants

General

The following paragraphs explain the terms used with regard to consistency assurance and show different SYNC/FREEZE configuration variants.

Terms

- **Bus cycle time**

Cycle in which the DP master (CP50M1) addresses all slaves once. In multi-master systems, all masters poll their slaves. The bus cycle time is configured and calculated in STEP 7 based on the baud rate, number, and type of slaves.

- **Sampling time**

Cycle in which the SYNPRO function block and the transmit and receive function blocks (on SIMATIC TDC CPUs) are calculated. The sampling time is configured in CFC.

Note

The bus cycle time and sampling time are independent.

- **Syncycle**

Syncycle is a multiple integer of the sampling time and can be configured at input CNX of function block SYNPRO. (Syncycle=CNX x sampling time).

A syncycle always begins with a sampling time. A sync command is always triggered by the SYNPRO function block in system mode at the start of a sampling time.

Configuration variant 1

Configuring variant 1 corresponds to most of the applications:

- Generating SYNC commands
- Data consistency is ensured at all slaves
- The synchronization cycle is at least twice the duration of the sampling time (CNX>1).
 - The length of the transmit frames (outputs) per slave may not exceed 32 bytes.
 - All transmit blocks and the SYNPRO function block must be configured for the same sampling time.
 - The SYNPRO function block must be configured chronologically before all transmit blocks (sequence of execution).
 - Output SOK of function block SYNPRO must be connected with the enable inputs of all transmit blocks (belonging to a slave group).
 - The bus cycle time must be less than the synchronization cycle minus 1 x the sampling time. At runtime, verify that output SOK goes to “1” once per syncycle. If not, increase the syncycle.

Example:

- Syncycle=3 x sampling time
- Bus cycle time=2 x sampling time
- Assumption: Function block SYNPRO starts calculations in the middle of the sampling time (before all transmit blocks)

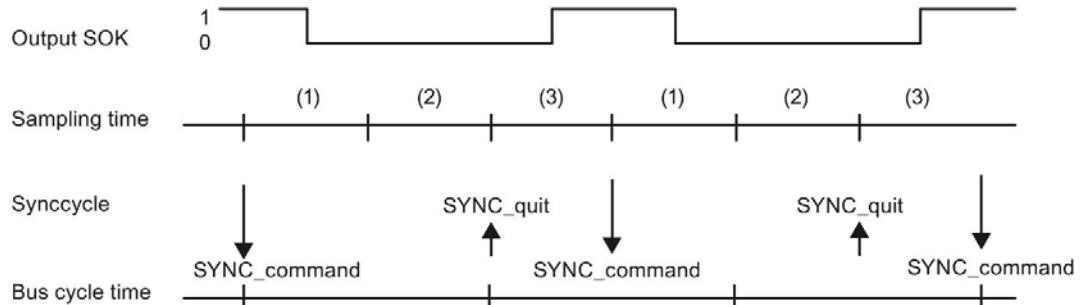


Figure 3-18 Timing diagram SYNC variant 1

After the SYNC command was triggered, the transmit blocks will be inhibited (SOK=0) for the duration of two sampling times (one bus cycle time). The transmit blocks are enabled (SOK=1) in the third sampling time after the SYNC command has been triggered.

Configuration variant 2

Configuration variant 2 provides the maximum SYNC performance:

- Generating SYNC commands
- Data consistency is ensured at all slaves
- Syncycle=sampling time (CNX=1)
 - The length of the transmit frames (outputs) per slave may not exceed 32 bytes.
 - All transmit blocks and the SYNPRO function block must be configured for the same sampling time.
 - High baud rate (>1.5 Mbaud); timing conditions can hardly be maintained at lower baud rates.
 - The bus cycle time may may not exceed 50 % of the sampling time.
 - The bus cycle time must also be of a length so that it expires between the start of a sampling time and the start of calculation at function block SYNPRO. This condition cannot be ensured and must be checked at runtime.

Example:

- Synccycle=sampling time
- Bus cycle time=0.3 x sampling time
- Assumption: Function block SYNPRO starts calculations in the middle of the sampling time (before all transmit blocks)

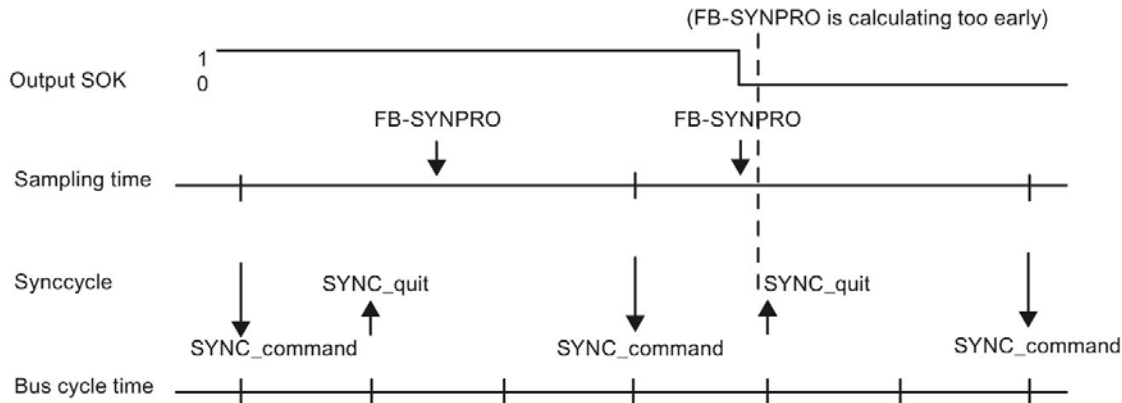


Figure 3-19 Timing diagram SYNC variant 2

Usually, the transmit blocks are always enabled (SOK=1). If the SYNPRO function block is calculated before SYNC has expired (on the right in the figure) due to time fluctuation, the transmit data are not updated and the values from the previous sampling time are transmitted instead. The syncycle and data consistency are not influenced by this action.

Notes on optimizing SYNC functionality:

In addition to a short syncycle, it is also necessary reduce jitter (time-related fluctuation) to minimum in the syncycle. The following precautions support this action:

- Irregular data traffic on the DP bus should be prevented: single-master mode; no temporary activation of additional stations.
- Interrupt tasks should not be configured on the same SIMATIC TDC CPU. Timeout of the sampling time is not allowed, as this would result in a SYNC command failure, or in an offset by a complete sampling time.
- Configure a high baud rate and short frame lengths (jitter is added up to the slave polling time).
- Configure the SYNPRO function block and all associated transmit blocks with T1=T0 (basic sampling time). With this setting, the SYNC command is always triggered along with the basic clock cycle interrupt that offers better timing precision compared to an interrupt that is initiated in the system mode.

Configuration variant 3

Configuration variant 3 corresponds with less commonly used FREEZE applications:

- Generating SYNC and FREEZE, or only FREEZE commands
- Data consistency is ensured at all slaves
- The syncycle is at least three times the length of the sampling time (CNX>1).
 - The length of the transmit or receive frame (inputs or outputs) may not exceed 32 bytes per slave
 - All transmit and receive blocks and the SYNPRO function blocks must be configured with the same sampling time (on the same CPU)
 - The SYNPRO function block must be configured as the last function block in the processing sequence
 - Output SOK of function block SYNPRO must be connected with the enable inputs of all (belonging to the slave group) transmit and receive blocks
- The bus cycle time must be less than the synchronization cycle minus 2 x the sampling time. At runtime, verify that output SOK goes to "1" once per syncycle. If not, increase the syncycle.

Example:

- Syncycle=4 x sampling time
- Bus cycle time=2 x sampling time
- Assumption:

Function block SYNPRO starts calculation in the middle of the sampling time (after all receive and transmit blocks)

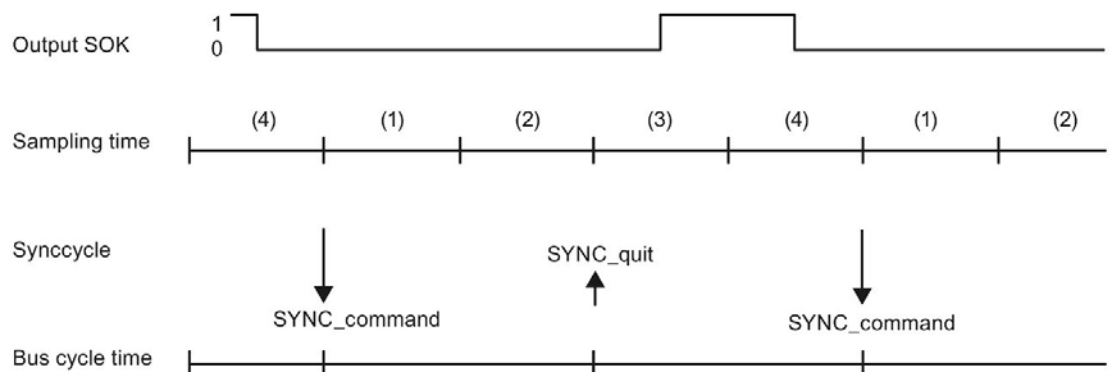


Figure 3-20 Timing diagram SYNC variant 3

After the SYNC command has been triggered, the transmit and receive blocks are inhibited (SOK=0) for the duration of three sampling times (one bus cycle time + one sampling time). The transmit and receive blocks are enabled (SOK=1) in the fourth sampling time after the SYNC command has been triggered.

3.6.5 Commissioning/diagnostics

3.6.5.1 Diagnostics function block

General

Master- or slave-specific diagnostics data from PROFIBUS DP can be output using the DPDIAG, DPSLDG, and DIAPRO function blocks.

For more information on diagnostic data, refer to the user documentation for the slaves.

Diagnostics data overview

Function block **DPDIAG**: Diagnostics overview

- The system diagnostics provides an overview as to which slave has reported diagnostics information.
- The 4 double words are bit coded,
- with each bit being assigned a slave and its PROFIBUS address in accordance with the following table.
- The active bit at the assigned slave indicates that this slave has reported diagnostics information.

Table 3- 5 Assignment of the system diagnostics/data transfer list to the slave PROFIBUS address

Output	Bit 16	Bit 15	Bit 14	...	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
DG1	15	14	13	...	4	3	(2)	(1)	(0)
	31	30	29		20	19	18	17	16
...
DG4	111	110	109		100	99	98	97	96
	-	-	(125)	...	116	115	114	113	112

Data transmission list:

- The data transmission list provides an overview of the slaves that were involved in data transmission within a configured time.
- The 4 double words (DL1 – DL4) are bit coded as for system diagnostics.
- The active bit set for the assigned slave indicates that it is currently used to transmit data.

Master status:

- Output of master-specific information:

Table 3- 6 Master-specific information

Output	Significance
MST	DP master status: Stop (40h), Clear (80h), Operate (C0h)
ID	Identification number: 815Eh for CP50M1

DPSLDG block: Slave diagnostics

- Output of slave diagnostics data.
- The SLA entry corresponds to the slave PROFIBUS address.
- The diagnostics data is dependent on the slave type.
- The first 16 bytes of slave diagnostics data are output.
- Additional slave diagnostics data can be output with SEL>1000.

For more information on slave-specific diagnostics data, refer to the user documentation for the respective PROFIBUS slaves.

Diagnostics data of DP slaves

Table 3- 7 Overview of the structure of the diagnostics data of DP slaves

Connection		
ST1	Status 1	Diagnostics to Standard 6 bytes
ST2	Status 2	
ST3	Status 3	
MPA	Master PROFIBUS address	
ID	Identification number	
D01 – D59	Device-specific diagnostics (refer to the user documentation of the respective PROFIBUS slave)	

Bits of status 1, 2, and 3

Table 3- 8 Significance of the status bits 1, 2, and 3

	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Status 1 (ST1)	S: Slave was parameterized by a different master.	S: Error in last parameter frame	M: Incorrect slave response	S: Requested function is not supported	S: Diagnostics entry in the specific diagnostics area	S: Configuration data inconsistent	S: Slave is not ready for data exchange	M: Slave is unavailable on the bus
Status 2 (ST2)	M: Slave not entered as "deactivated"	(not used)	S: Slave has received a Sync command	S: Slave has received a Freeze command	S: Response monitoring activated	S: 1 (fixed)	S: Diagnostic data must be fetched	S: Parameterization and configuring required
Status 3 (ST3)	S/M: Diagnostics data cannot be transmitted	-	-	-	-	-	-	-

- **M:** Master detects diagnostics data
- **S:** Slave detects diagnostics data

Master PROFIBUS address (MPA)

- PROFIBUS address of the master that parameterized this slave.
If this slave is not parameterized, then FFh is set.

Identification number (ID)

- Identifies the slave type.

All additional diagnostics data is slave-specific.

Generally followed (standard DP slave) by the diagnostics blocks: Device-related, identification-related and channel-related diagnostics. Not all slave-specific diagnostic blocks must be available.

Each block is preceded by a header byte. Bits 7 and 8 identify the diagnostics block:

Table 3-9 Significance of bits 7 and 8 in the header byte

Bits 7 and 8 in the header byte	Significance
Bit 7, 8= 00	Device-related diagnostics
Bit 7, 8= 01	Identification-related diagnostics
Bit 7, 8= 10	Channel-related diagnostics

Bits 1 to 6 define the following:

- For device- and identification-related diagnostics, the length of the diagnostic block including the header byte, value range 2...63.
- For channel-related diagnostics, the identification number, value range 0...63.

3.6.5.2 Error class (ECL) and error code (ECO)

Outputs ECL, ECO

Significance of the outputs ECL, ECO at function block @PRODP:

- **Error Class>0:** Error pending. Function block @PRODP transmits a communication error (lit "LED" on CP50M1).

These connections are only of secondary significance, as the corresponding communication errors can be read from the diagnostics buffer. You may be requested to submit these connection values if you contact the Hotline for support on fatal errors.

3.7 MPI coupling

Properties

MPI (Multi Point Interface) is the standard communication protocol in SIMATIC S7/M7. Data is exchanged via multi-master bus with a maximum of 126 nodes.

In SIMATIC TDC, MPI is not only used to connect the CFC for commissioning and testing a configuration, but also for communication with WinCC and SIMATIC OPs.

MPI coupling is used in combination with the communication utilities service (FB-SER) and S7 Communication (FB-S7OS).

Hardware

Hardware requirements for MPI coupling:

- Rack
- CPU
- CP50M1 module with corresponding interface configured in HW Config for MPI, at CP50M1 only X01)
- MPI cable (included in the scope of delivery of the programming device)

HW Config

The CP50M1 communication module and the corresponding interface must be configured as MPI in HW Config. The internal MPI address must be specified for ES.

Function block @MPI

Exactly one @MPI central coupling block must be configured for each interface. The @MPI function block initializes and monitors the MPI coupling.

For more information on the configuration of an MPI coupling, refer to the following sections:

- Sec. Communication utility service (Page 232)
- Sec. AUTOHOTSPOT
- Sec. Communication with WinCC (TCP/IP) (Page 278)

3.8 PNIO communication

3.8.1 Overview

NOTICE
Line/ring connection
Make absolutely sure that there is no ring interconnection with the physical connection of the PROFINET network (only exception is when the MRP domain has been configured accordingly).

Introduction

A @PNIO PROFINET central block must be connected to the PROFINET interface for CPU555. Process data can be both sent and received. TCP/UDP communication can also be configured.

This configuration model addresses configuration engineers and shows how to implement a coupling from SIMATIC TDC to a PROFINET station.

This configuration model contains the SIMATIC TDC function blocks and shows the principles of their application. The functional scope of the configuration model is restricted to essential steps to get you started as soon as possible. You may always expand functionality and/or the hardware components. However, you must always observe the specifications in the respective function block documentation. The designations used for data blocks, flags, variables, etc. are arbitrary.

Channel modes

PNIO communication is possible with the following channel modes:

- Handshake
- Refresh

PNIO integration

From the user's point of view, receiving and sending process data via PROFINET looks just like PROFIBUS DP.

The CTV_P and CRV_P blocks are used to send and receive.

However, note that you need to configure the @PNIO central block instead of @PRODP.

Note

Fault detection

To detect all errors on the PROFINET bus and be able to react to them, you need to cyclically check the error outputs of the send and receive blocks in the user program.

Additional information

- System Manual Industrial Ethernet / PROFINET - Passive Network Components (<http://support.automation.siemens.com/WW/view/en/84922825>)
- System Manual PROFINET system description (<http://support.automation.siemens.com/WW/view/de/19292127/0/en>)
- System Manual Industrial Ethernet/PROFINET (<http://support.automation.siemens.com/WW/view/en/27069465>)

3.8.2 Configuration model

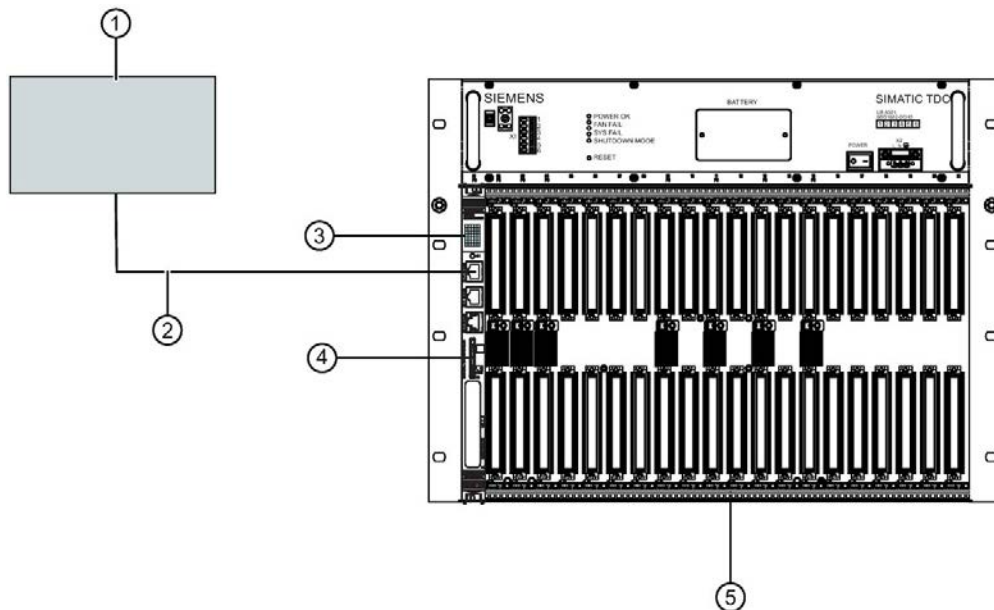
Requirements

The structure of these configuration instructions reflects the chronological order of steps to take for creating the entire configuration. However, these instructions only represent a recommendation and are not mandatory.

Appropriate knowledge of SIMATIC Manager (including HW Config and CFC) and about the configuration of SIMATIC TDC is presumed.

Hardware

The following devices or components were selected and for the configuration model and positioned as follows :



- ① PROFINET station
- ② PROFINET IO cable
- ③ CPU module CPU555
- ④ Program memory card MMC
- ⑤ Rack UR6021

Figure 3-21 Configuration model

3.8.3 Configuring SIMATIC TDC

Configuring SIMATIC TDC with a CPU555 and PROFINET station.

The @PNIO function block is used to initialize and monitor the local PROFINET coupling. The data interface for this coupling is located in local RAM of the CPU.

Note**One @PNIO block for each CPU555**

The @PNIO function block may only be configured once for each CPU555, since there is only one CPU for a PROFINET coupling.

Multiple configuration of the @PNIO function block is detected during initialization and triggers an entry in the communication error field.

Note**Isochronous mode**

Only one send block and one receive block can be configured for isochronous mode.

In the case of multiple configuration, the CPU555 reports an error after startup.

Address parameters

Specifications at address connection AT, AR with CTV or CRV blocks

You specify the use of the PNIO RT/IRT protocol with the entries at address connection AT or AR of the send/receive blocks.

The address levels 1 and 2 must be specified in addition to the channel name. The two address levels are separated in the notation by a period ".".

Input sequence:

"Channel name.Address level 1.Address level 2"

- **Channel name:**
 - Max. 8 characters
 - ASCII characters except "dot" and @
 - Channel names of all send and receive blocks that access the PNIO interfaces of a CPU555 must be different
 - The channel name has no particular significance for PNIO
- Append a dot "." to the channel name entry

Address level 1

- PROFINET RT
Device number of the communication partner (IO device)
Example: DEV01.01 or DEV01.01.1
- Isochronous
The string "PIP" (Process Image Partition) is used.
Example: BELNAM.PIP

Note

IO device after an IE/PB Link PN IO

An IO device after an IE/PB Link PN IO is addressed via its unique device number.

Append a dot "." to the address level 1 entry (optional)

Address level 2 (optional)

- Consists of up to two characters, of which the second is of no significance to CP50M1.
- With PROFINET RT, the byte ordering is determined by a digit.
 - 0: Bytes are not rotated (default setting)
The data is transmitted in "Motorola format" (most significant before least significant byte).
 - 1: Bytes are rotated
The data is transmitted in "Intel format" (least significant before most significant byte).
This setting can be used for communication partners whose internal data management is based on the Intel format (e. g. SIMATIC TDC)

Annotations:

Direct device communication:

- The data of a device are read in the "physical" order, i.e. in the order in which the modules are plugged in and in the order in which they are sent from the device.
- The channels for the devices are updated cyclically every 1-100 ms depending on the load of the CPU.
- Outputs located in the process image partition cannot be written as a device via the addressing.
Inputs can be read both via the process image partition as well as via the device addressing.
- With addressing as a device, the length of the data must **exactly** match the number of bytes that the device provides (visible in HW Config).

Isochronous communication:

- With isochronous modules, the data is read as a process image. The order is determined by the configured address.

Gaps are read as well. The addresses should therefore be entered in the process image partition PIP1 if possible starting at address 0 with no gaps, otherwise the run time of the send and receive blocks is increased.

- The process image partition is updated in isochronous mode, then the "PNIO clock cycle (Ready to Receive)" interrupt is triggered.
- With addressing as process image partition, the length of the data (frame length NBY) must correspond to the highest address + 1 in the process image partition.

Example: Configuring via the process image partition (isochronous)

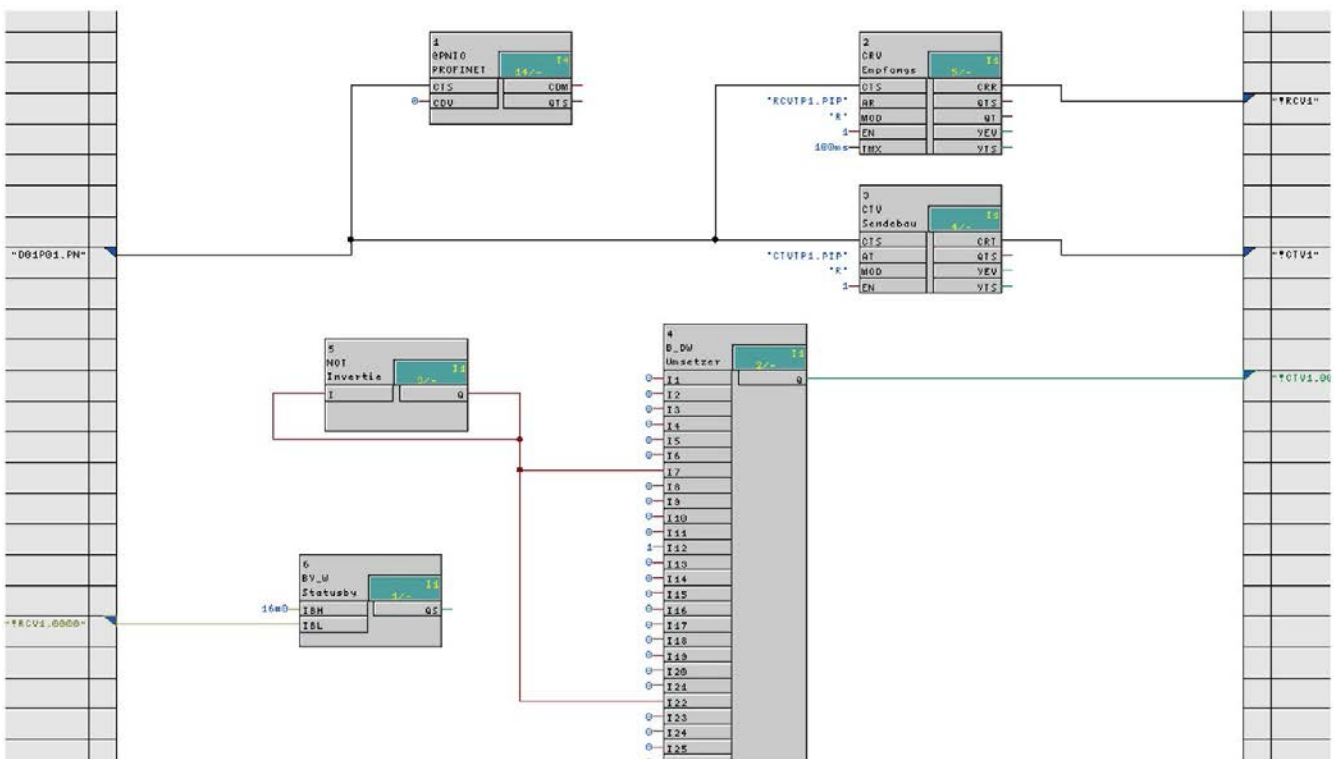


Figure 3-22 Example process image partition

Example: Configuring via the device number

If the data of a device is to be read, the device number (1-128) is provided instead of the process image partition.

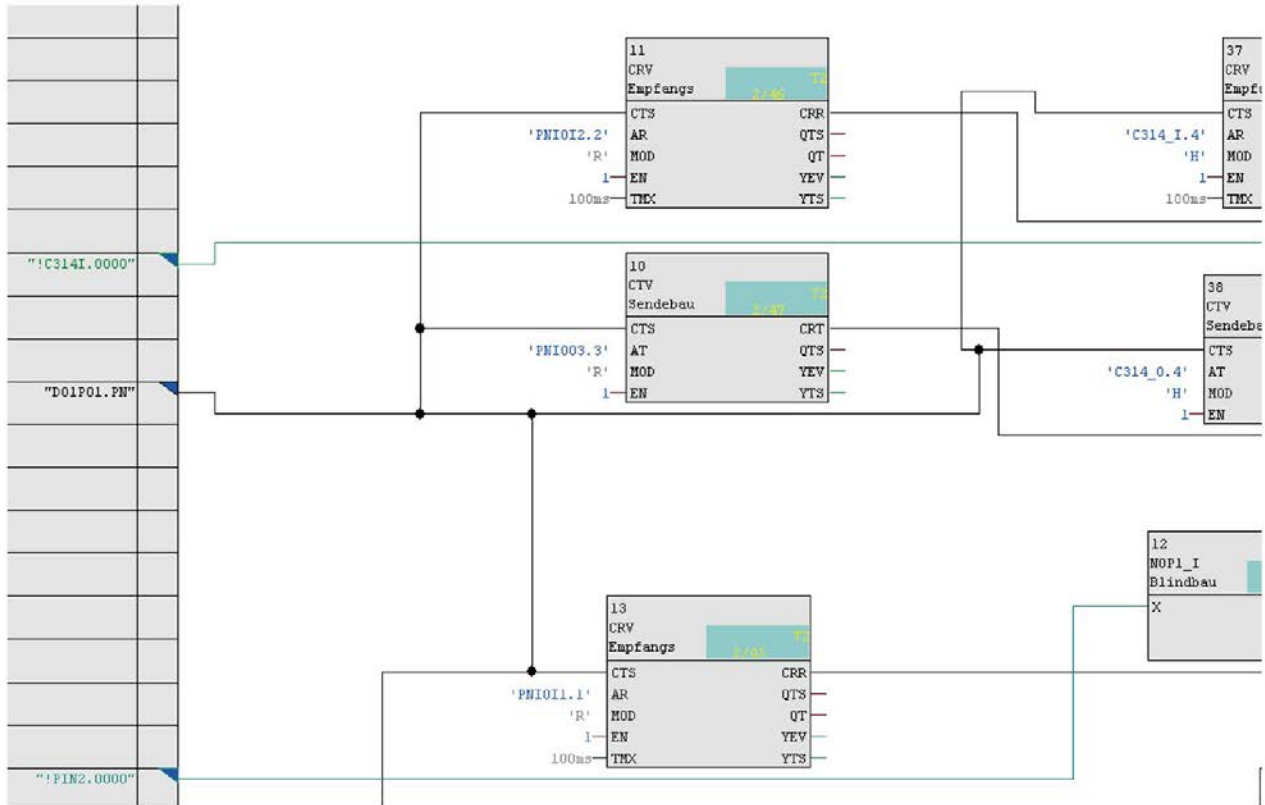


Figure 3-23 Example device number

3.8.4 Use isochronous I/Os

Note**Isochronous mode with CPU555**

The "PNIO interrupt (receive data ready)" signal has high jitter and is therefore not suitable for synchronization of the basic clock cycle (T0). Therefore, the PNIO send clock is used as the basic cycle clock. However, this means that the received data have not been read at the start of T1 and must be read with a delay.

To achieve this, the basic cycle clock must be delayed in HW Config or in the user program.

Note

- In order to guarantee error-free isochronous mode, you should not call any data record services for isochronously operated IO devices and there should be no configured alarms if possible.
 - Changing to a different processing method (e.g. output-input processing) may also make sense in order to reduce/prevent error messages.
-

The EVA model should be observed when using isochronous I/O. The EVA model is a program processing model:

- Input: Read in process image partition with CRV or CRV_P
- Processing: In the user program
- Output: Output process image partition with CTV or CTV_P

A task needs to be performed in isochronous mode and the calls of the CTV_P and CRV_P blocks need to be executed with address stage 1 "PIP" in this task.

To do this, either an alarm task must be started with "PNIO cycle (receive data ready)" or T0 must be synchronized to "PNIO cycle (send cycle)". In this case, the inputs are read made available to the user program.

For the synchronization of T0 to "PNIO cycle (send cycle)", also set a suitable delay time so that the receive data is available when T0 is started. If the selected delay time is too low, you receive an error message at the CRV_P block informing you that the receive data is not yet available.

You can derive a guideline value for the delay time to be selected with the following formula:

$$\text{Delay time} = 300 \mu\text{s} + \text{size of process image partition in bytes} \times 1 \mu\text{s}$$

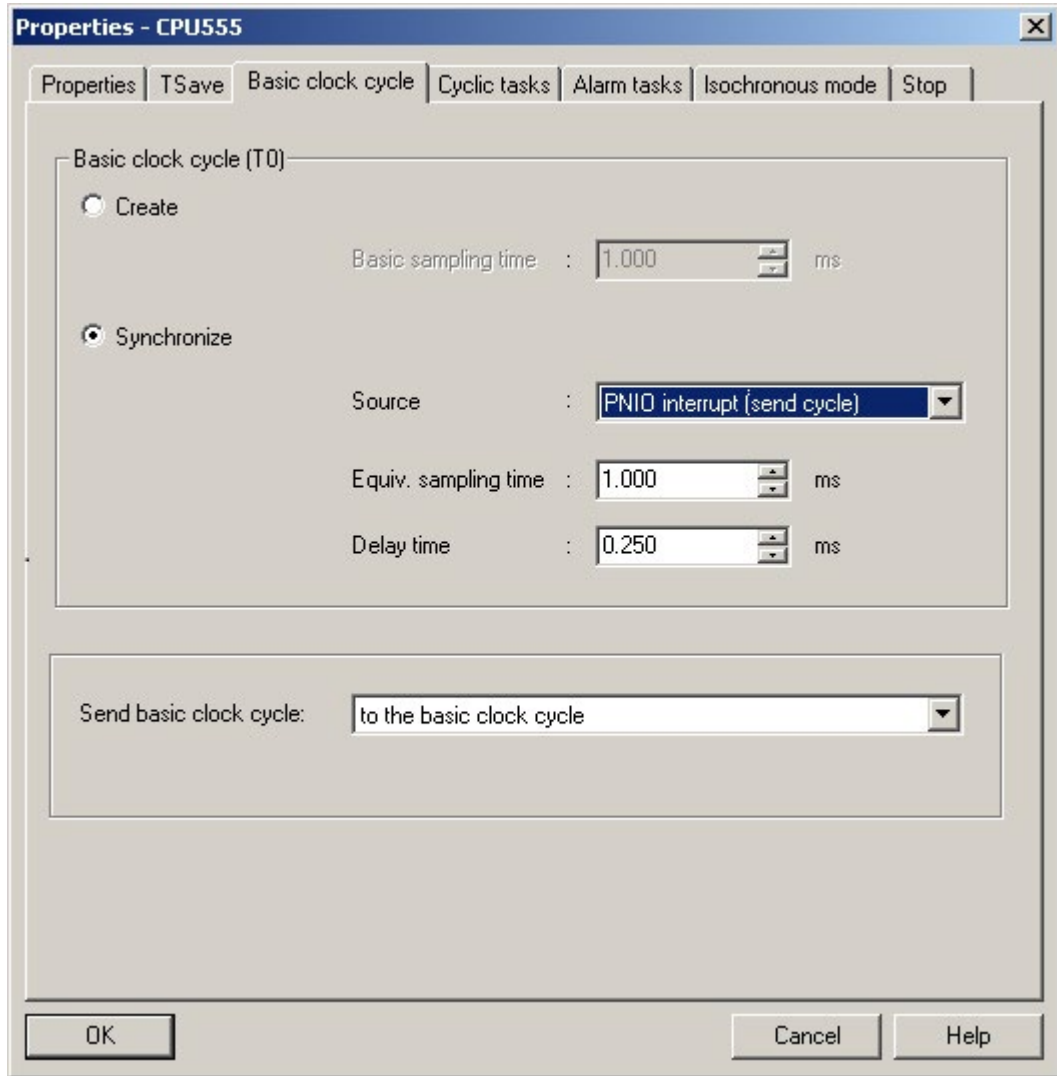


Figure 3-24 Synchronize CPU to the PROFINET clock cycle

Also, select the IO system number, and the process image partition in the Isochronous mode tab.

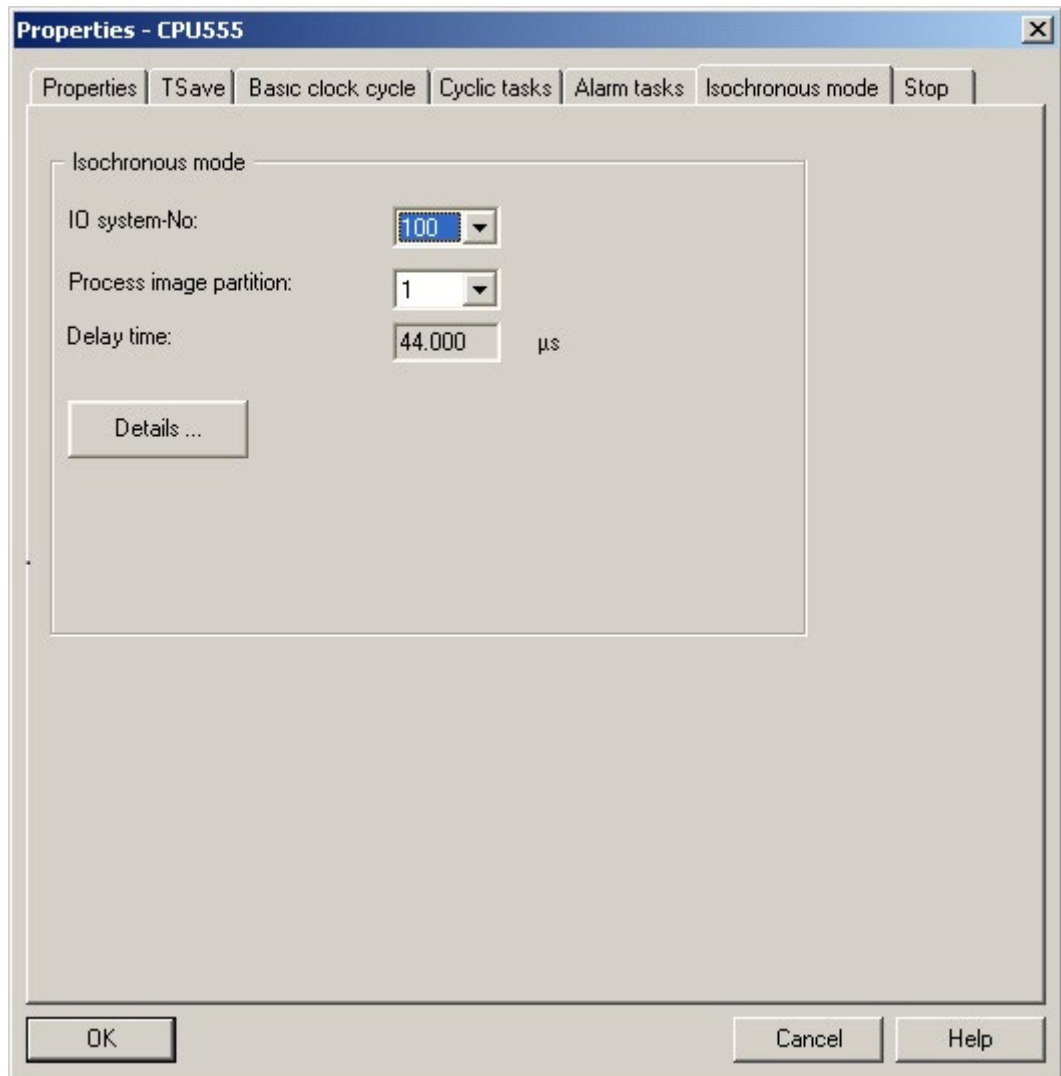


Figure 3-25 Setting the isochronous mode

3.9 Table function

3.9.1 Introduction

The table function in SIMATIC TDC provides you with the option of implementing and using tabular values in a configuration. Configure the TAB and TAB_D function blocks accordingly at the SIMATIC TDC end. Tabular values of data type REAL are managed with function block TAB, whereas values of data DINT are managed with function block TAB_D. The user provides the tabular values.

The table function can be configured in three modes:

- **Manual mode:** The tabular values are entered directly at the block via online interface (e.g. CFC in test mode), or transferred to the block using teach-in from the program.
- **Automatic mode: Communication:** the tabular values are transferred via communication interface (TCP/IP, DUST1, S7 via P bus).
- **Automatic mode: Memory card:** the tabular values are loaded to the memory card and read from this card.

Note

Note that it is only possible to change between the "**Manual mode**" and "**Automatic mode: Communication**" and between "**Manual mode**" and "**Automatic mode: Memory card**".

The validity of tabular values entered or transmitted will be checked. The address of the table is displayed at the "TAB" output.

The tabular values are managed twice, i.e. in two tables. The "validated" (=active) table is used for all arithmetic operations of the configuration. The "invalid" (=inactive) table is used to manage value changes. All tabular values you have changed are initially transferred to the invalid table. Once the inactive table is activated, the new tabular values are mirrored to the second table. The previously active table automatically loses its validity. This means that the new tabular values are available in both tables.

Both tables can be saved to the SAVE area with backup battery in order to prevent data loss (I/O SAV=1 during initialization).

Note

You can find a detailed description of the TAB and TAB_D function blocks in the online help.

You can find a detailed description of the WR_TAB function block under "Function block WR_TAB (Page 159)".

3.9.1.1 "Manual mode" overview

The following figure illustrates basic procedures in "Manual mode":

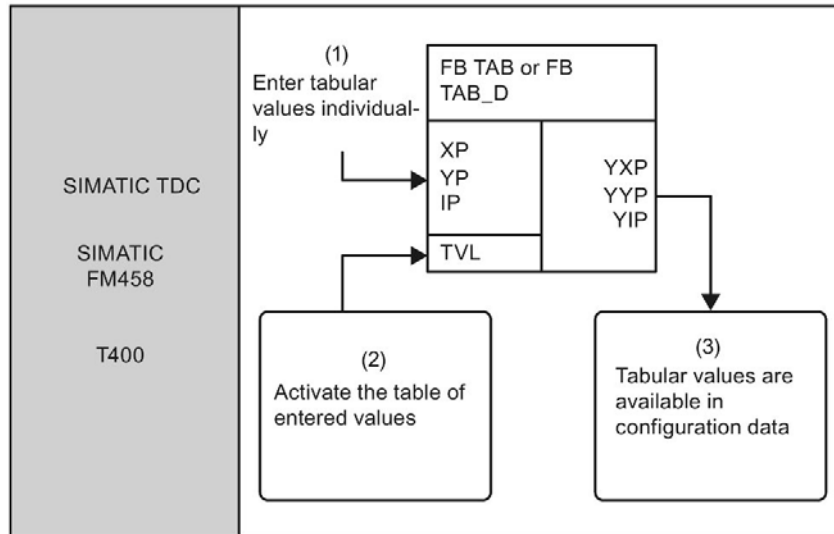


Figure 3-26 Basic procedures in "Manual mode"

Details of the "Manual mode" are available in section "AUTOHOTSPOT".

3.9.1.2 "Automatic mode: Communication" overview

In "Automatic mode: Communication", tabular values can be transmitted using the following communication variants:

- S7 via P bus for SIMATIC FM 458 (requires an additional configuration of WR_TAB on controller side)
- TCP/IP (tabular values can be transmitted from a SIMATIC TDC module to another one using the CTV and CRV FBs)
- DUST1 (tabular values can only be transferred via DUST1 interface)

The tabular values are transmitted by means of data frames.

The following figure illustrates basic procedures in "Automatic mode: Communication" for transmission of tables via P bus from an S7 controller to a SIMATIC FM 458 application module:

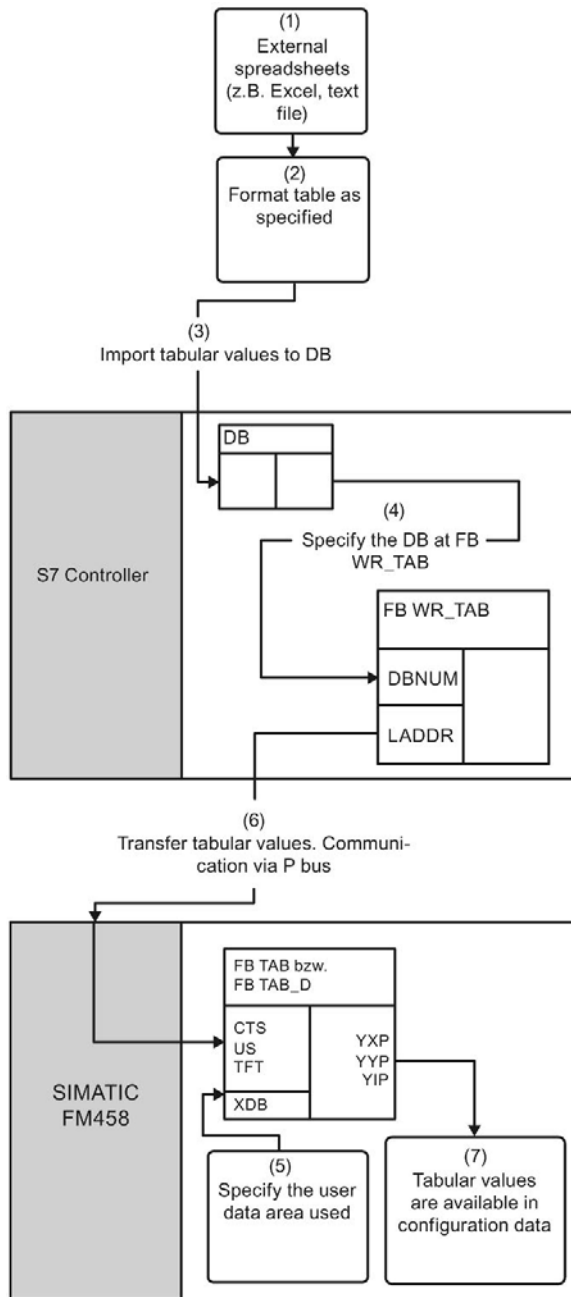


Figure 3-27 Basic procedures in "Automatic mode: Communication (via P bus)

For more information on the "Automatic mode: Communication" for transmission of tables via P bus from an S7 controller to a SIMATIC FM 458 application module, refer to chapter "Automatic mode: Communication (Page 164)".

3.9.1.3 "Automatic mode: Memory card" overview

In "Automatic mode: Memory card", you can load the tabular values to the memory card with the help of PC software (D7-SYS additionalComponentBuilder) and transmit this data during initialization of the module.

For more information on the "Automatic mode: Memory card", refer to chapter "Automatic mode: Memory card (Page 183)".

3.9.1.4 Function block WR_TAB

Symbol

WR_TAB		
Block activation	BO EN TABTEL W	Number of data blocks required to transfer the complete DB content
Request to write a new table	BO REQTAB CNTTEL W	Number of data blocks already transferred
Request to write tabular values to the data block	BO REQDB STATUS W	Current processing state
Last data block for the table	BO LASTDB ERROR W	Alarms as required
Logical module address	W LADDR DONE W	DONE status parameter: Transmission completed
Data record number for the read and write data record	BY RECNUM	
Data block number	W DBNUM	
TIMEOUT for receiving the acknowledge frame from the FM module	DW TFT	

Short description

Function block WR_TAB serves for transmitting tables from an S7 controller to a SIMATIC FM 458 application module. The tabular values (valid data types are REAL and Double Integer) are stored in a data block. WR_TAB transfers the values to the TAB or TAB_D function blocks that manage the tabular values internally on the SIMATIC FM 458 application module.

Configure function block WR_TAB on controller side. The tabular data is transmitted via P bus from an S7-400 controller to a SIMATIC FM 458 application module. The transmission always includes all tabular values that are available at the DB that is specified at the DBNUM input.

Connections

The following table lists the various connections and their data type, as well as a description of the connections:

Parameters	Declaration	Data type	Description
REQTAB	INPUT	BOOL	REQTAB = 1: Request to write a new table
REQDB	INPUT	BOOL	REQDB = 1: Request for write access to the tabular values stored in the data block
LASTDB	INPUT	BOOL	Last DB for the table
LADDR	INPUT	WORD	Logical address of the SIMATIC FM 458 application module
RECNUM	INPUT	BYTE	Data record number for the read and write data record
DBNUM	INPUT	WORD	Data block number of the DB that contains the tabular values.
TFT	INPUT	DWORD	TIMEOUT in ms for receiving acknowledge frames from the SIMATIC FM 458 application module.
TABTEL	OUTPUT	WORD	Number of data blocks required to transfer the complete DB content
CNTTEL	OUTPUT	WORD	Number of data blocks already transferred to the FM module
STATUS	OUTPUT	WORD	Displays the current processing / transmission status. 0: Table transmission is deactivated 1: Table transmission is activated. Transmitted segment with tabular values from a DB (waiting for next segment) 2: Transmission of tabular values from a data block is not yet completed.
ERROR	OUTPUT	WORD	If an error occurs while processing the function, the return value contains an error code
DONE	OUTPUT	BOOL	DONE=1 status parameter: Transmission completed

The following errors may occur and are indicated at the ERROR output:

Error code	Explanation	Remedy
0xB210	OK	-
0xB211	Invalid logical address of the module	Specify a valid module address at input LADDR.
0xB212	Invalid data record number	Enter the tabular values in ascending order in the DB.
0xB213	Invalid data format in table	Tabular values must be of data type REAL for TAB and of data type DINT for TAB_D.
0xB214	The data format of the new data record does not match that of the previously transferred data records	Ensure that all tabular values have the same data format.
0xB215	FM 458 not responding	Check the communication coupling and configuration.
0xB216	Table too large	Run a segmented transmission of the table, i.e. distribute tabular values to several DBs, or write new (additional) tabular values the DB and transmit these.
0xB217	Table is incomplete (X / Y values)	Complete the table; a Y value must be available for each X value.
0xB218	REQTAB was reset during processing	Repeat transmission of the tabular values.
0xB219	REQDB was reset during processing	Repeat transmission of the tabular values.
0xB21A	Invalid DB number	Specify a valid DB number.
0xB21B	TIMEOUT when receiving the acknowledge frame	Check the communication coupling and configuration. Repeat transmission of the tabular values.
0xB21C	Invalid processing state	Check the WR_TAB configuration.

Moreover, errors at SFC58 or SFC59 are indicated at the ERROR output.

3.9.2 Manual mode

Application

The "Manual mode" represents the simplest way of inserting tabular values into a configuration. However, manual input or teach-in from the program are relatively time consuming methods.

Entering tabular values

Once TAB or TAB_D has been properly configured, you can successively enter the tabular values. Start by specifying the table size at input NP, i.e. the number of value pairs (=points). Input SAV of the function block must be set to 1 if the table is to be saved to the SAVE area.

You can then enter the tabular values. Start by specifying index point *i* at input IP of the value pair to be entered. Enter the X and Y value of the point at inputs XP and YP. After having entered each value pair, set input WR from 0 to 1 to apply the values. Increment the index at input IP before you enter the next point and then enter the values for this point. This procedure is repeated until all values have been entered.

It is not necessary to observe a specific order for entering the points.

The number of points entered must match the data at input NP.

All entries made in the course of this procedure are applied to the inactive table of the function block and are only available after being activated in the configuration. Set input TVL to 1 to activate the inactive table with the values entered.

You can now continue to edit the inactive table, whereby changes made will not be available until you reactivate the table.

Querying the tabular values

To enable the output of the tabular values you entered, specify the index of point *i* that is to be displayed and set input RD from 0 to 1 after having completed your entries at input IP. The tabular values of point *i* are now displayed at the YXP (X value) and YYP (Y value) outputs and the index of point *i* is returned at output YIP.

Configuration

For the "Manual mode", you only need to configure the TAB and/or TAB_D FBs, depending on whether tabular values of data type REAL and/or DINT have to be managed. Each table may only contain values of the same data type. If you want to manage several tables of different data types, configure a TAB or TAB_D FB for each table.

The TAB and TAB_D function blocks should be configured with a sampling time greater than or equal to 32 ms. The following I/O settings must be made:

AUT = 0 (automatic mode deactivated)
NP = [declaration of the table size]
XP = [input of X values]
YP = [input of Y values]
IP = [input of the value pair to be changed]
TVL = 1 (for activating the table after all values have been entered)
WR = 1 (for writing the value pair entered to the table)
RD = 1 (for displaying the value pair specified under IP at the YXP and YYP outputs)

Note

If "Manual mode" is active and the CTS I/O is set to "0" during initialization (CTS=0; AUT=0), you cannot switch to the "Automatic mode: Memory card" (CTS=0; AUT=1).

However, you can switch to "manual mode" (CTS=0; AUT=0) if the CTS connection is set to "0" during initialization and "Automatic mode: Memory card" is activated (AUT=1). You can now edit the table that is stored on the memory card in "Manual mode".

If you return to the "Automatic mode: Memory card" (CTS=0; AUT=1) after having completed your changes, this mode will no longer have any effect as it is only active in the initialization process.

If a communication interface is configured for the CTS I/O, you may always switch from "Manual mode" to "Automatic mode: Communication" and vice versa.

3.9.3 Automatic mode: Communication

3.9.3.1 Application with S7 controller and SIMATIC FM 458 application module

Transferring tabular values

The following requirements must be met for the successful transmission of tables:

- Function blocks TAB and/or TAB_D are configured at the FM 458 application module in accordance with the configuration defaults for "Automatic mode: communication". (Details are available in the section "Configuration for the S7 controller and SIMATIC FM 458 application module").
- The X and Y values of a table in a DB must always be present alternating. Each X value must be assigned a Y value in order to obtain an even number of values in a data record.

Set inputs REQTAB and REQDB at WR_TAB to 1 to start transmission. The tabular values of the DB that is specified at input DBNUM of WR_TAB are now transmitted.

The actual number of data blocks transmitted is always indicated at the CNTTEL output of WR_TAB.

The TABTEL output of WR_TAB indicates the number of data blocks you need to transmit the entire content of the DB to the SIMATIC FM 458 application module.

If all tabular values are available in the specified DB, or if this was the last segment transmitted for a table that does not "fit" completely into a DB, set input LASTDB of the WR_TAB to 1 prior to the start of transmission. This signals the end of transmission to the SIMATIC FM 458 application module. The STATUS output of WR_TAB is now reset from 2 to 0.

Note

Transmission always encompasses all tabular values contained in the DB that is specified at input DBNUM of WR_TAB.

Table too large for a DB

If the table is too large for a data block, transmit the tabular values by means of segmented frames. Proceed as follows:

Write the first table section to the DB and then transmit the data as described above. The LASTDB input of WR_TAB remains at 0. The STATUS output of WR_TAB is set to 2 for the duration of transmission and then switches from 2 to 1 once the segment has been transmitted.

IN the next step, overwrite the already transmitted tabular values in the DB with the set of values. On completion, reset input REQDB of WR_TAB from 0 to 1 to activate transmission of the next segment.

Repeat this procedure until all tabular values have been transmitted.

Set input LASTDB of WR_TAB from 0 to 1 at the end of the last segment transmission. This signals the end of transmission to the SIMATIC FM 458 application module. The STATUS output of WR_TAB is now reset from 2 to 0.

Note

You may also span the table to several different DBs, provided sufficient user memory resources are available. In this case, you only need to specify the matching DB number at input DBNUM of WR_TAB for each frame transmitted. However, make sure that the DBs are transmitted in the correct sequence to ensure that all table values are transmitted in ascending order.

Transmission cycle

The time it takes to transmit the tabular values depends on the following factors:

- Number of tabular values
- Size of data blocks
- Sampling time of TAB and TAB_D
- WR_TAB execution time

A frame containing 56 tabular values is transmitted in each cycle from the controller to the SIMATIC FM 458 application module.

The transmission time for a table can then be calculated as follows:

Duration of the transmission = [number of table values / 56] x cycle time of the slowest FB (i.e. TAB, TAB_D, or WR_TAB)

The time it takes to transmit the data via the P bus is not relevant for this estimation, as this transmission cycle is generally less than 1ms and the TAB and TAB_D function blocks are usually are configured for sampling times longer than 32 ms.

Time requirements will increase if the table is distributed to several data blocks, as the time it takes for the manual changes mentioned above is added to the transmission time that is calculated based on the formula specified above.

3.9.3.2 Configuration for the S7 controller and FM 458 application module

Configure the following function blocks for interconnecting an S7 controller with a SIMATIC FM 458 application module via P bus:

- SIMATIC FM 458 application module:
 - TAB (for data type REAL) and/or
 - TAB_D (data type DINT)
 - @CPB (central block for P bus coupling)
- S7 controller:
 - WR_TAB

Each table may only contain values of the same data type. If you want to manage several tables of different data types, configure a TAB or TAB_D FB for each table.

WR_TAB serves for the transmission the tabular values from SIMATIC DB to the TAB and TAB_D function blocks. The tabular values are transmitted in data frames. Once the last data frame was transmitted, the TAB or TAB_D FBs are automatically informed that all tabular values have been transmitted and that the table is to be activated. WR_TAB receives a response frame that indicates whether or not the table was successfully activated. If successful, the table address is returned at output TAB of FB TAB or TAB_D.

TAB and TAB_D

Configure TAB and TAB_D as follows:

Set a sampling time greater than or equal to 32 ms. The following connection settings must be made:

- CTS** = [name of the configured communication interface]
- AUT** = 1 (automatic mode activated)
- US** = [channel name.address level1] (address specified for receiving)
- MOD** = [transmission mode] (H=Handshake; R=Refresh; S=Select; M=Multiple)
- TFT** = [monitoring time in milliseconds] (maximum missing frame timeout during reception of tabular values)
- NP** = [specifies the maximum table size]

Note

If a communication interface is configured for the CTS connection, you may always switch from "Automatic mode: Communication" to "Manual mode" and vice versa.

WR_TAB

The following connection settings should be configured at WR_TAB:

- LADDR** = [specifies the logical address of the SIMATIC FM 458 application module]
- RECNUM** = [specifies the data record number for the read and write channel. Must be identical with "address level1" at the US connection of TAB or TAB_D.]
- DBNUM** = [specifies the data block number]

3.9.3.3 Inserting tabular values in the data block

The tabular values to be transmitted to a SIMATIC FM 458 application module must be available in a data block (DB). The DB must be programmed on controller side.

There are two ways of generating a DB that contains the required tabular values:

- Creating a new DB in STEP 7 and entering the tabular values manually in the "LAD/STL/FBD" application
- Importing tabular values from a table (e.g. MS Excel) as external source in STEP 7

Entering tabular values in manual mode

This is the simplest method of providing tabular values in a DB, involving manual input of the initial and actual tabular values manually in a newly generated DB in the "LAD/STL/FBD" application. The following section describes the necessary steps to take.

Note

The start value can be defined individually for every table. It is only used if no actual value is specified for the relevant tabular value.

The actual value is made available as tabular value in the configuration. The required tabular values should be specified here.

(1) Generating a new DB in STEP 7

Start by creating a new DB in STEP 7. For this purpose, select the "Blocks" folder in the corresponding S7 program and then select "Insert new object > Data block" command from the shortcut menu.

The following figure shows the procedure:

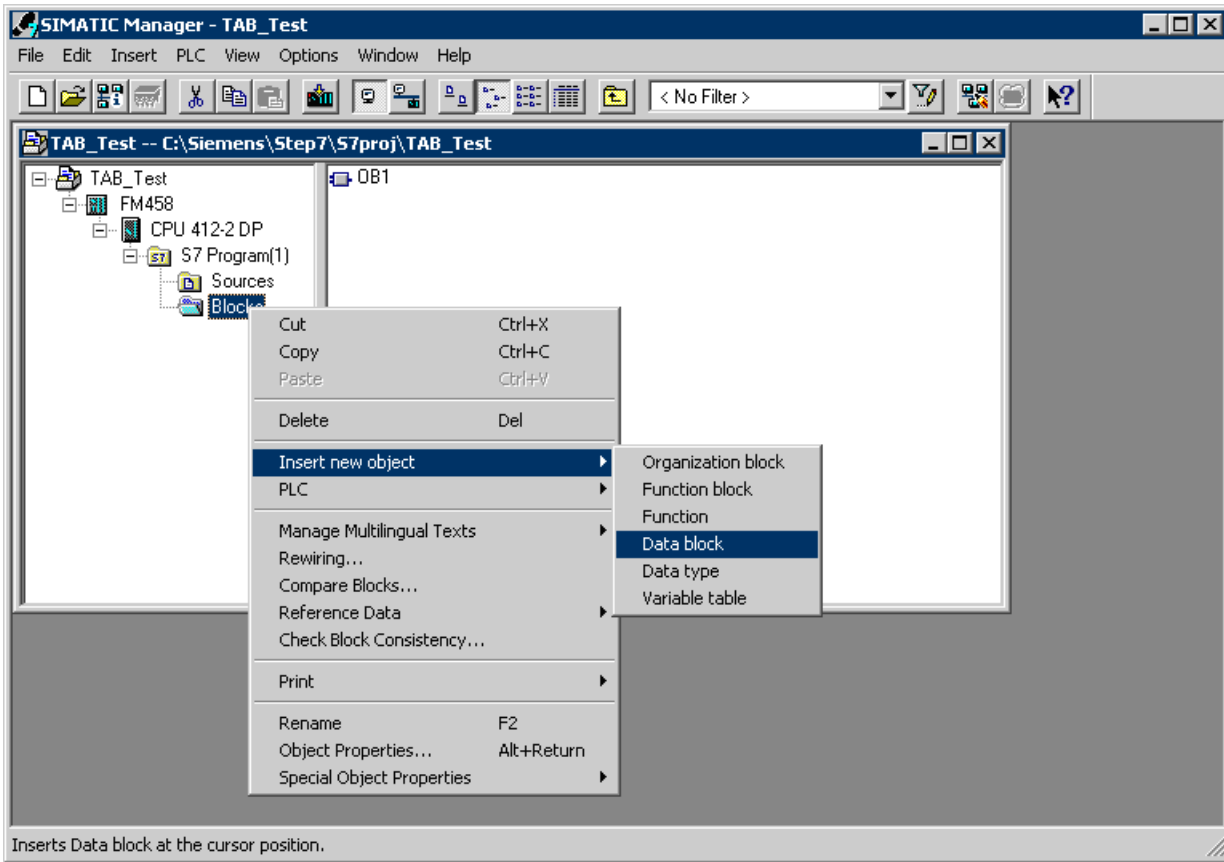


Figure 3-28 Creating a new data block in STEP 7

(2) Opening the new DB

The next step is to open the new DB in the "LAD/STL/FBD" application with double-click. "DB Editor" is the editing tool that is only used to create a "Data block".

The following diagram illustrates the selection when opening a new DB:

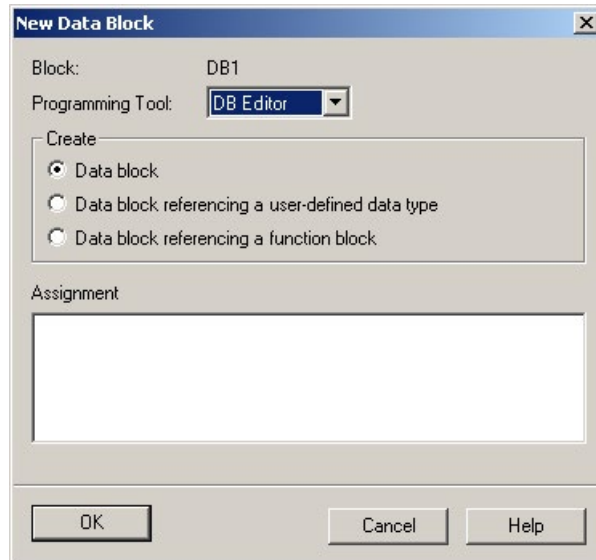


Figure 3-29 Options for creating a new DB

The opened, new DB is illustrated in the following diagram:

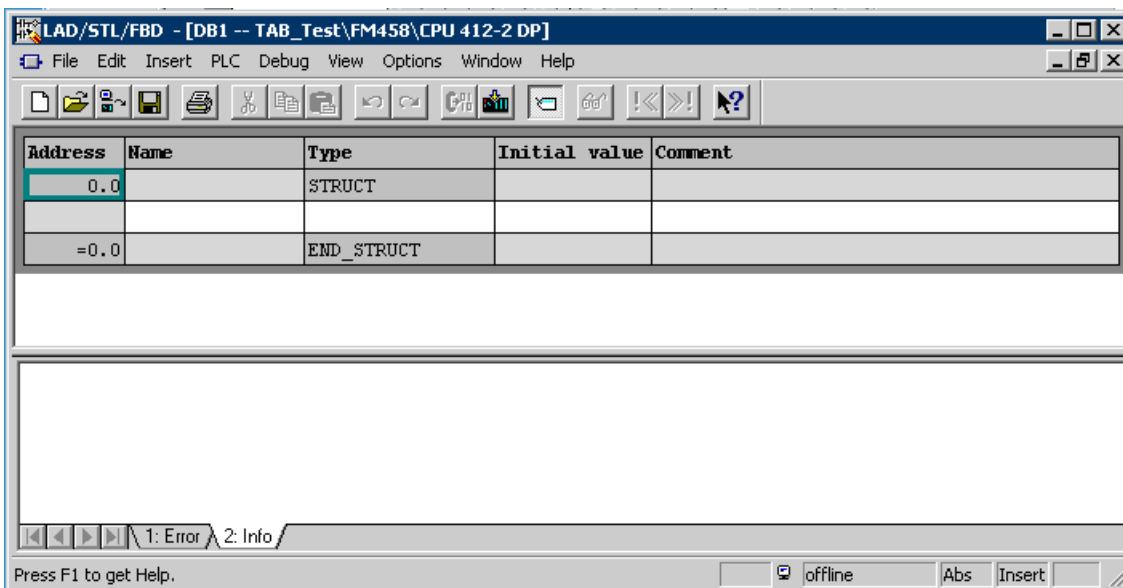


Figure 3-30 New DB created in the "LAD/STL/FBD" application

(3) Entering the tabular values

You can now enter the required tabular values. Make sure that you enter the X and Y values in alternating mode.

First thing, enter the data type (REAL or DINT) used in the table. Always enter the name "Data type", "WORD" type, with start value "W#16#1" for data type REAL and "W#16#2" for data type DINT.

Next, enter the name, data type ("Type" column) and value ("Initial value" column) for each tabular value.

The following figure illustrates the procedure for entering tabular values of data type REAL:

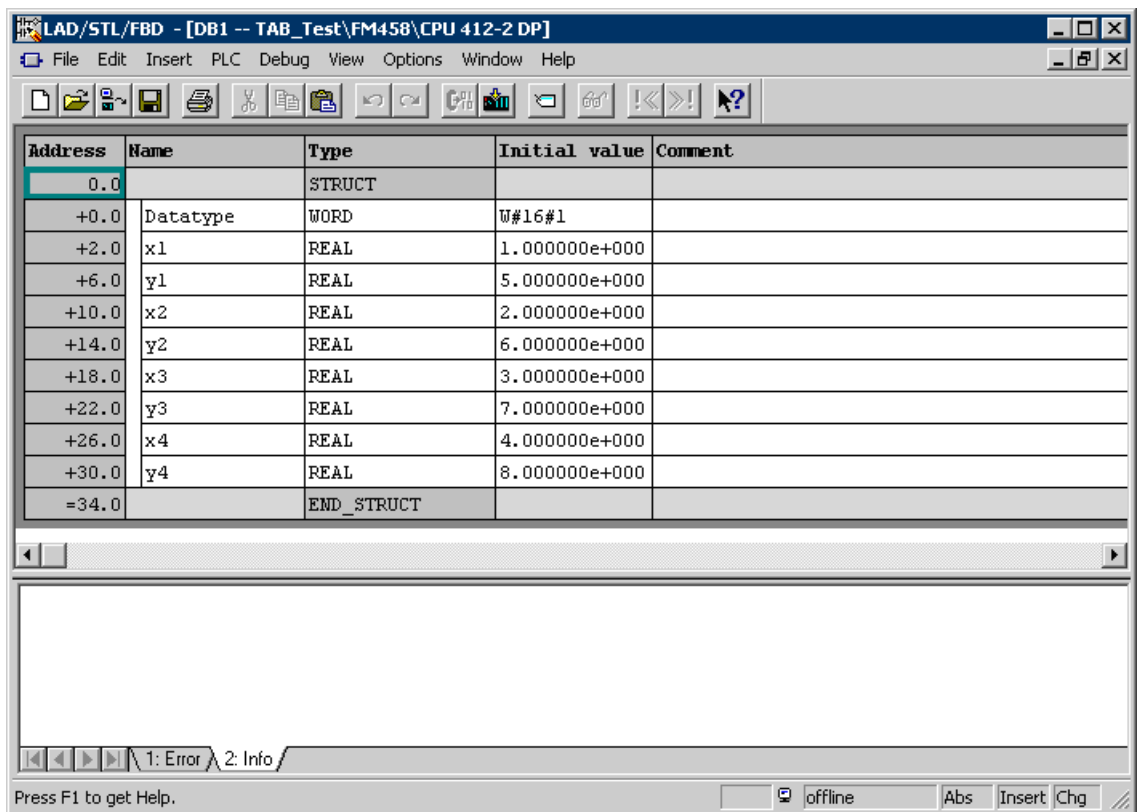


Figure 3-31 Tabular values entered in the "LAD/STL/FBD" application in manual mode

Note

Since the table may only contain values of the same data type, you can specify an ARRAY as a more efficient method of input. This method saves you from having to specify the data type over and over again.

For information on the procedures for generating entries of the ARRAY type, refer to the Online Help of the "LAD/STL/FBD" application and in particular to "Help on STL".

(4) Saving the DBs

After having entered all tabular values, you can select "File > Save" to save the DB.

The tabular values are now available for transmission in the DB.

Importing tabular values

The tabular values to be made available in the DB can also be imported from an external source, e.g. an MS Excel table. However, observe the following aspects for error-free import:

- The source file of the table must have a specific format
- The source file must be implemented as external source file in STEP 7
- A new DB is created based the external source file

The following paragraphs describe the points to observe and the steps to take for the import.

Table format

- A table (e.g. created in Excel) that is to be imported to the DB must conform with a specific format syntax:
- The table must contain a header with information with regard to the DB name and the version.
- Specify the data related to the structure and data type of the tabular values
- end then enter tabular values (as start values).
- Make sure that you always declare the X and Y values in alternating mode.
- Save the table with the *.AWL file name extension.
- The table can now be used as external source file.

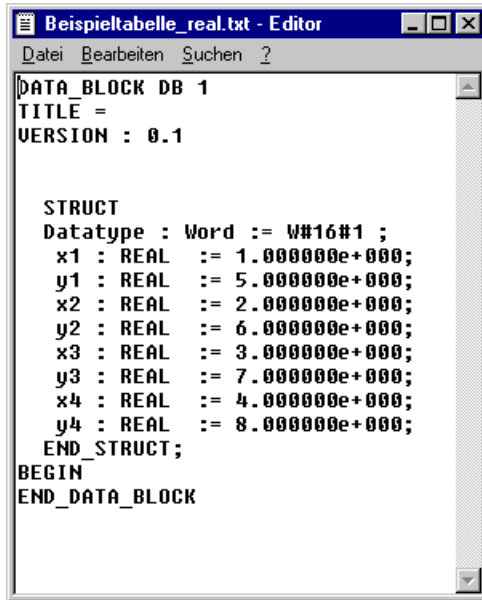
Note

You may declare any *start value* for each tabular value. It is only used if no actual value is specified for the relevant tabular value.

The tabular values are exclusively defined as *start values*. *Actual values* are not used.

This procedure considerably reduces the file size and memory space requirements.

The following figure illustrates an example of a table with four X and four Y values of data type REAL:



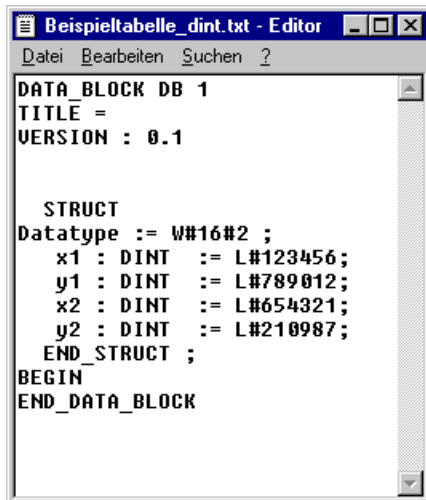
```
Beispieltable_real.txt - Editor
Datei Bearbeiten Suchen ?

DATA_BLOCK DB 1
TITLE =
VERSION : 0.1

STRUCT
Datatype : Word := W#16#1 ;
x1 : REAL := 1.000000e+000;
y1 : REAL := 5.000000e+000;
x2 : REAL := 2.000000e+000;
y2 : REAL := 6.000000e+000;
x3 : REAL := 3.000000e+000;
y3 : REAL := 7.000000e+000;
x4 : REAL := 4.000000e+000;
y4 : REAL := 8.000000e+000;
END_STRUCT;
BEGIN
END_DATA_BLOCK
```

Figure 3-32 Example of a table with values of data type REAL

The following figure illustrates an example of a table with two X and two Y values of data type DINT:



```
Beispieltable_dint.txt - Editor
Datei Bearbeiten Suchen ?

DATA_BLOCK DB 1
TITLE =
VERSION : 0.1

STRUCT
Datatype := W#16#2 ;
x1 : DINT := L#123456;
y1 : DINT := L#789012;
x2 : DINT := L#654321;
y2 : DINT := L#210987;
END_STRUCT ;
BEGIN
END_DATA_BLOCK
```

Figure 3-33 Example of a table with values of data type DINT

From Excel to STL

The following sections shows an example of the procedure for converting a table in Excel format to the table format that you need.

The example in following figure illustrates the step-by-step formatting of a file in accordance with specifications of the required table format.

	A	B	C
1	x-Wert	y-Wert	
2		1	5
3		2	6
4		3	7
5		4	8
6			
7			
8			
9			
10			
11			
12			
13			

Figure 3-34 Example of a table in MS Excel format

(1) Header

Start by entering the necessary header. Insert five rows at the start of the table and enter the following data:

- DATA_BLOCK DB 1 [DB number]
- TITLE = [enter as required]
- VERSION : 0.1 [version definition]

The following figure shows the Excel table with inserted header:

	A	B	C
1	DATA_BLOCK DB 1		
2	TITLE =		
3	VERSION : 0.1		
4			
5			
6	x-Wert	y-Wert	
7		1	5
8		2	6
9		3	7
10		4	8
11			
12			
13			

Figure 3-35 Example of an Excel table with inserted header

((2) Inserting the structure and tabular values

Insert the structure of the tabular values, including the values and data type definition. Insert two rows plus a start and end row for each pair of values. Enter a further row at the start for the definition of the data type used.

The start of the structural data is displayed in the start row by the "STRUCT" entry. Specify the data type used in the table in the next row ("W#16#1" for data type REAL, "W#16#2" for data type DINT).

Specify the structure and tabular values for each value pair and then enter the X and Y values in alternating mode. Specify the tabular values in accordance with the current data type definition (in this case, REAL). The end of the structure definition is indicated in the final row by the "END_STRUCT;" entry.

Complete your entries by defining the data component of the actual values ("BEGIN" and "END_DATA_BLOCK"). As the tabular values are already contained in the start values of the structure definition, the explicit input of specific actual values can be dispensed of.

The following figure shows the Excel table with the structure definition and tabular values:

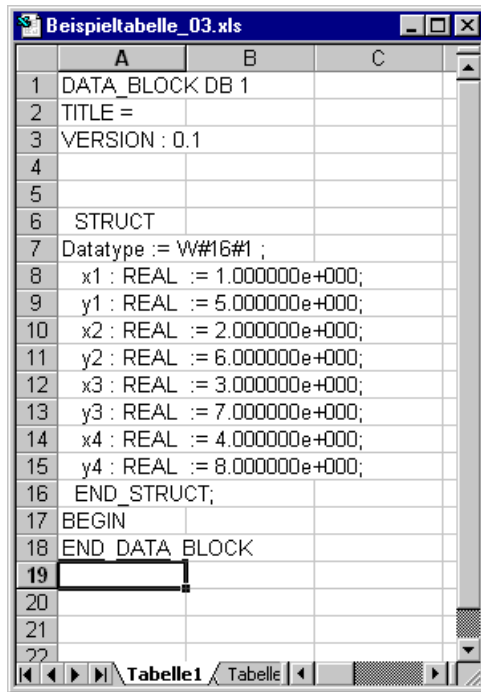


Figure 3-36 Example of an Excel table containing the structure definition and tabular values

(3) Saving the table to an STL file

To complete the task, you only need to save the properly formatted file to a text file with ***.AWL** extension. Select **"File > Save As..."** in MS Excel. In this dialog, select the **"formatted text (space separated) (.prn)"** file type and then name the sample table and save it to a path of your choice.

The following figure shows the "Save as..." window in MS Excel and the corresponding selection:

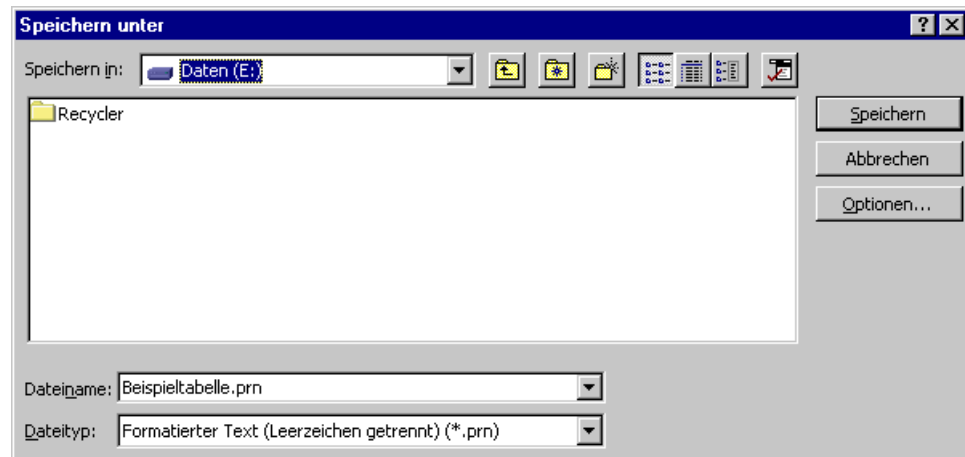


Figure 3-37 Example of an Excel table saved to a text file (*.prn)

Once the file has been saved, you only need to change the file type from *.prn to *.awl. This file can then be opened in any text editor.

The following figure illustrates the sample table in STL file format, opened in the standard Windows text editor:

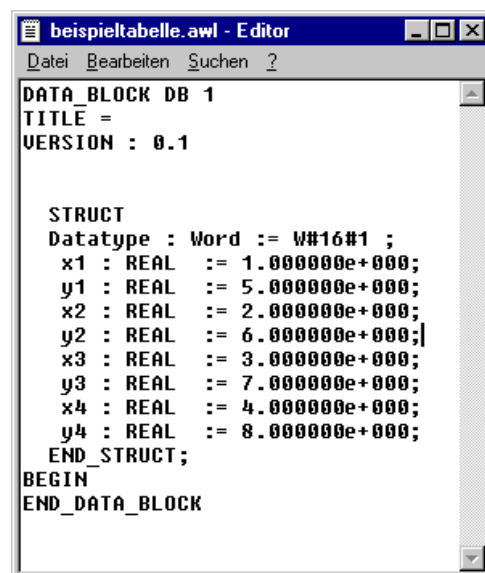


Figure 3-38 Sample table saved to an *.awl file and opened in the text editor

In STEP 7, this file can only be used as external source file for a DB.

Implementing the table as source file

This section describes the step-by-step implementation of an externally created table in a DB, based on the "EXAMPLETABLE.AWL" sample file you created previously.

Note

Along with the specification of tabular values, you particularly need to keep an eye on DB naming, as the DB will be generated based on the name you specified in the file.

In the above sample file, "DB1" is set as DB name in the first row.

You now have to insert an external source into the STEP 7 configuration by selecting "Sources" in the S7 program. After having selected "Sources", right-click in the window pane on the right side to open the shortcut menu. In his dialog, insert an external source as new object.

The following figure shows the procedure:

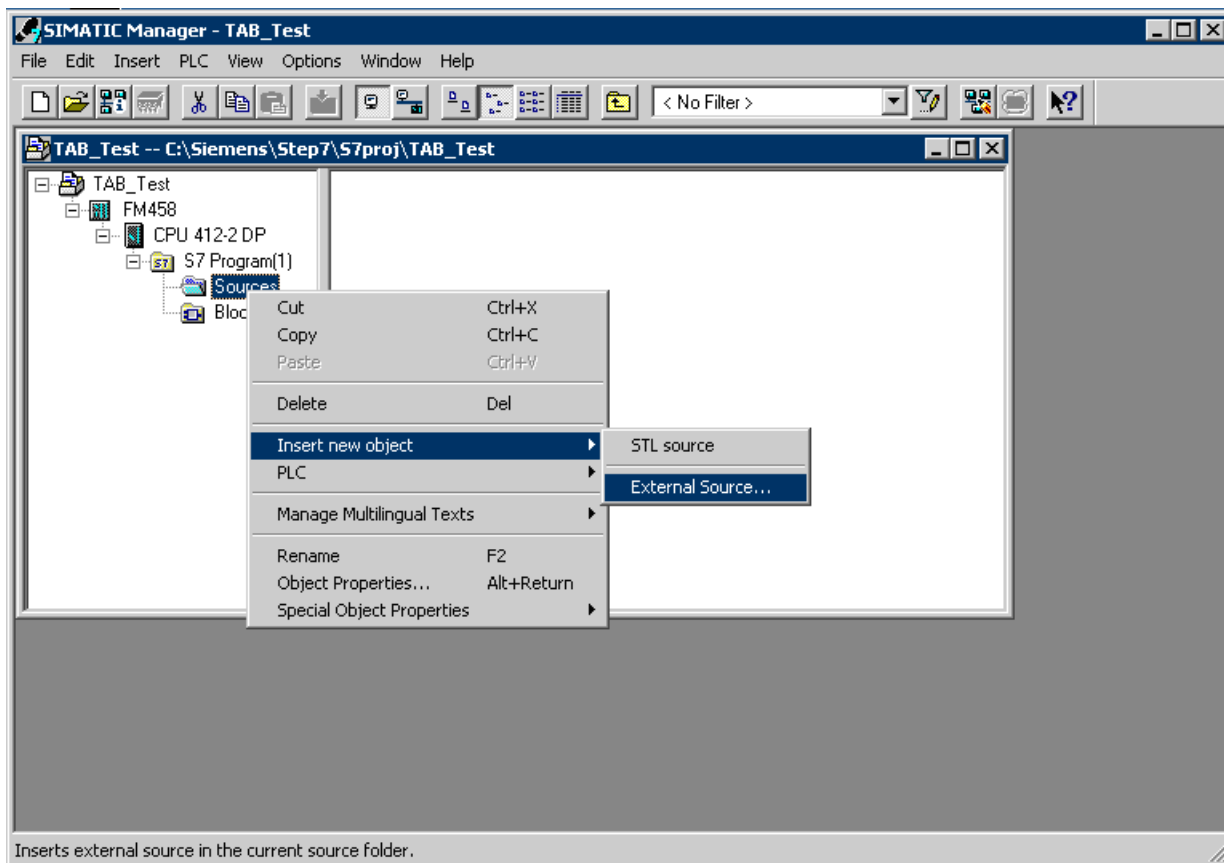


Figure 3-39 Inserting an external source in STEP 7

Select the STL file you generated previously as the source file. The following figure shows the file selection window:

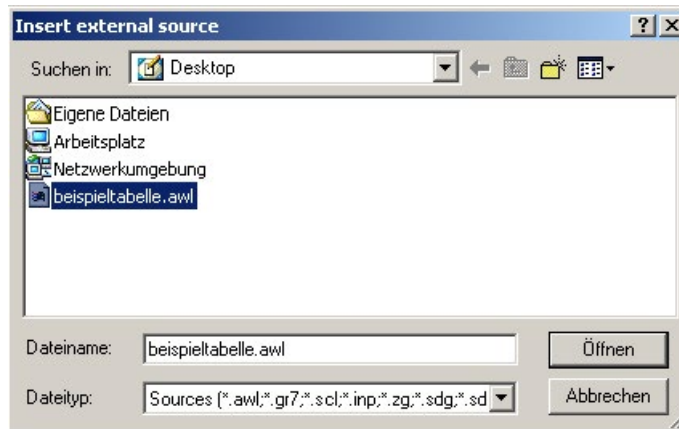


Figure 3-40 Selecting the file for insertion into STEP 7 as external source

The selected file is opened (here: EXAMPLETABLE.AWL). The file is now available as source file in the configuration under "Sources". The file can be selected re-opened in this dialog.

The following figure shows the sample file that is under "Sources", including the corresponding shortcut menu:

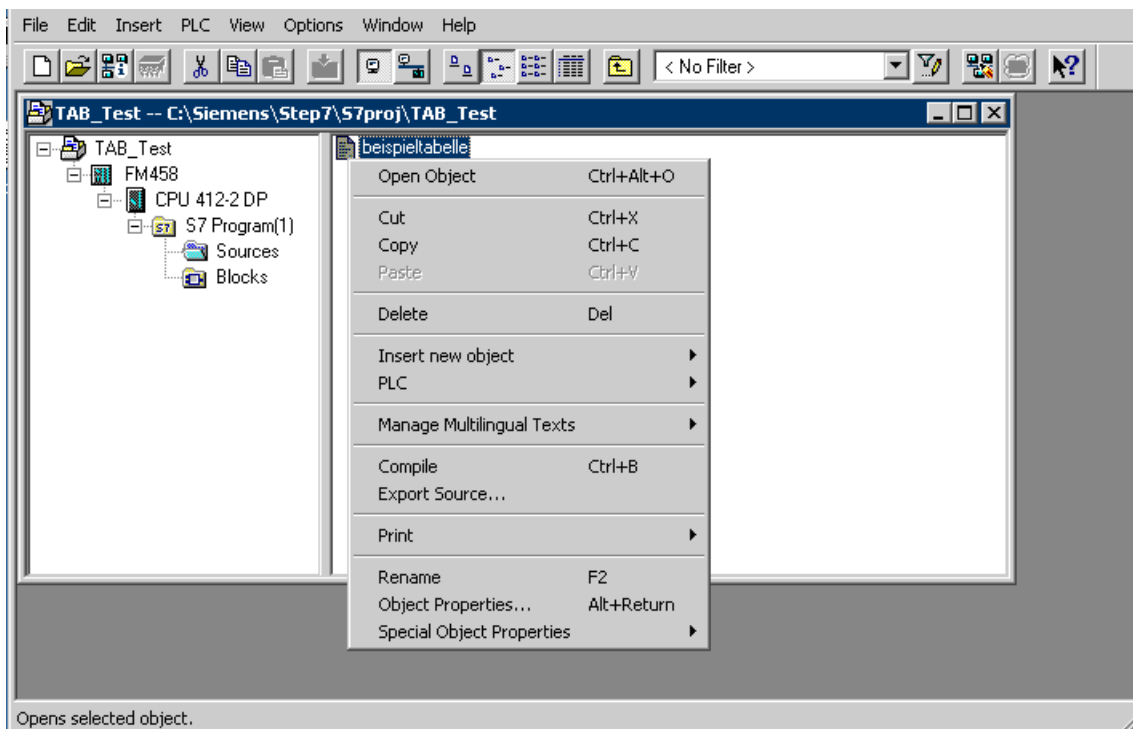


Figure 3-41 Generated source file in STEP 7

Once you opened the file, you can edit it in the "LAD/STL/FBD" program and you only need to compile it by selecting "File > Compile".

The following figure shows the procedure:

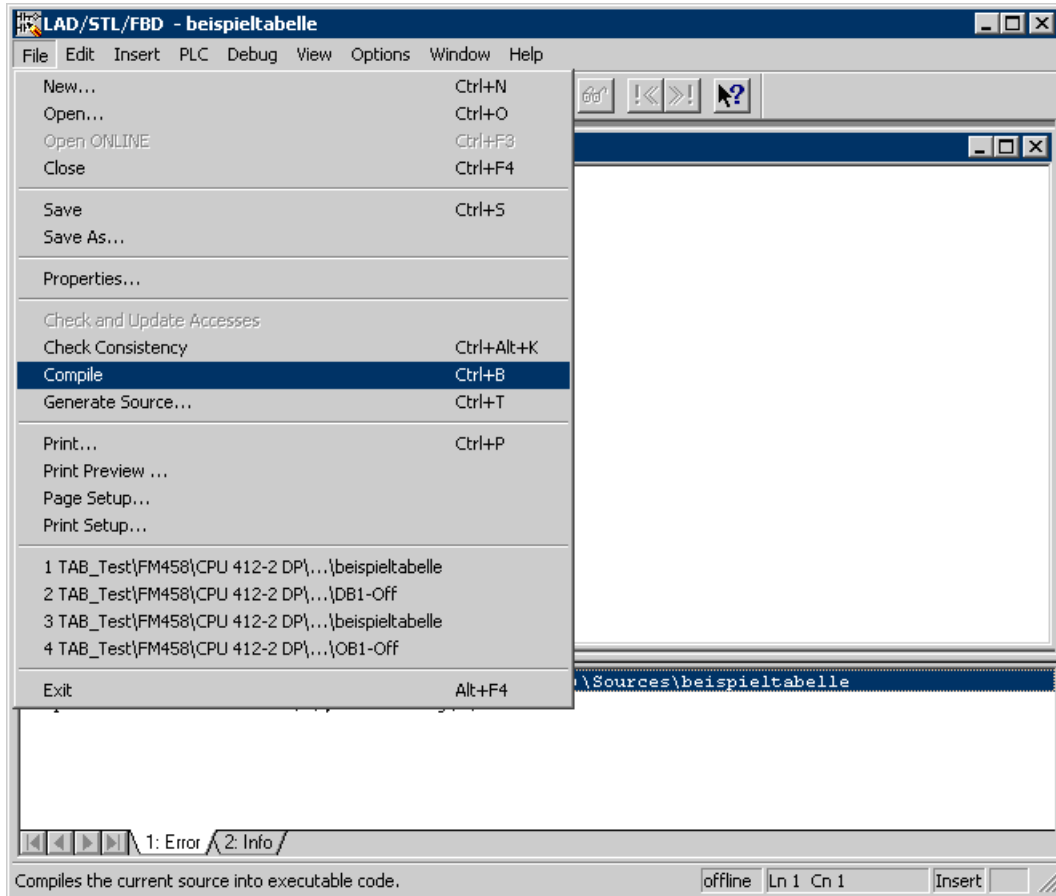


Figure 3-42 Compiling the source file in the "LAD/STL/FBD" application

After the file has been successfully compiled, a new DB is available in the configuration. The name of the DB corresponds to the name specified in the header row of the file.

The following figure illustrates the new DB generated under "Blocks" in the STEP 7 configuration:

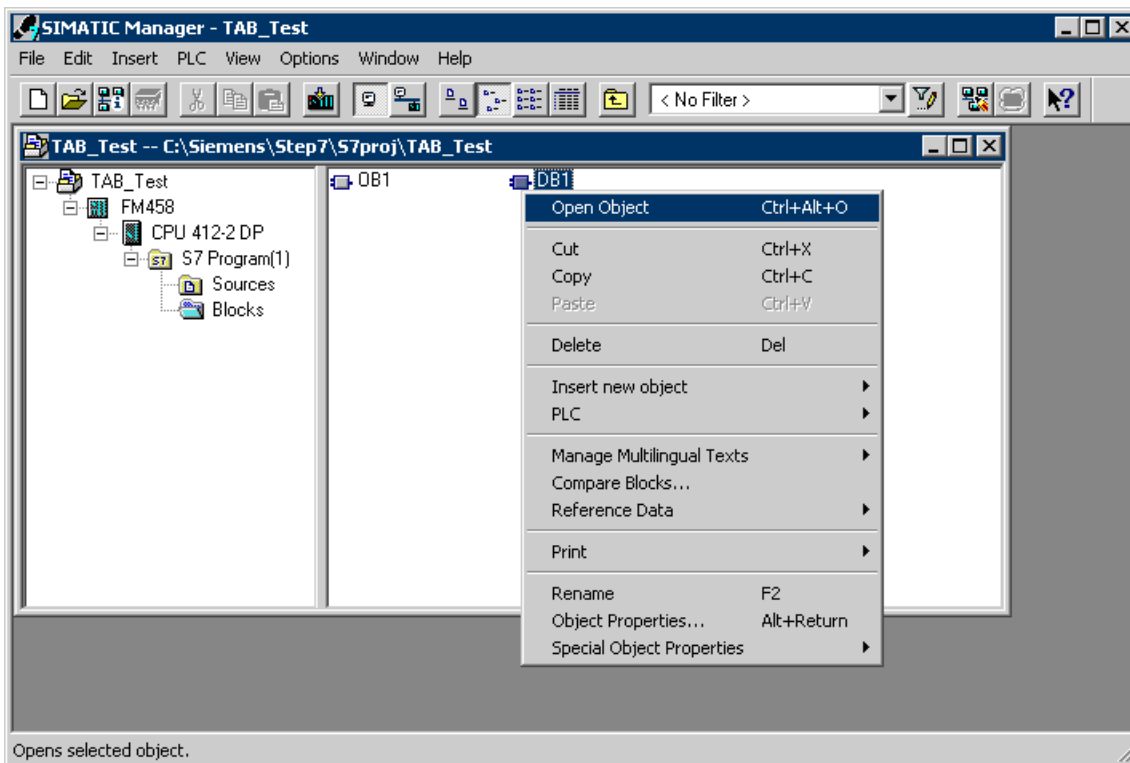


Figure 3-43 DB generated by compilation of the source file

You can open the DB in the "LAD/STL/FBD" application to check its content. Select the "Data view" command from the "View" menu to display the start values and actual values.

The following figure shows the content of the opened DB:

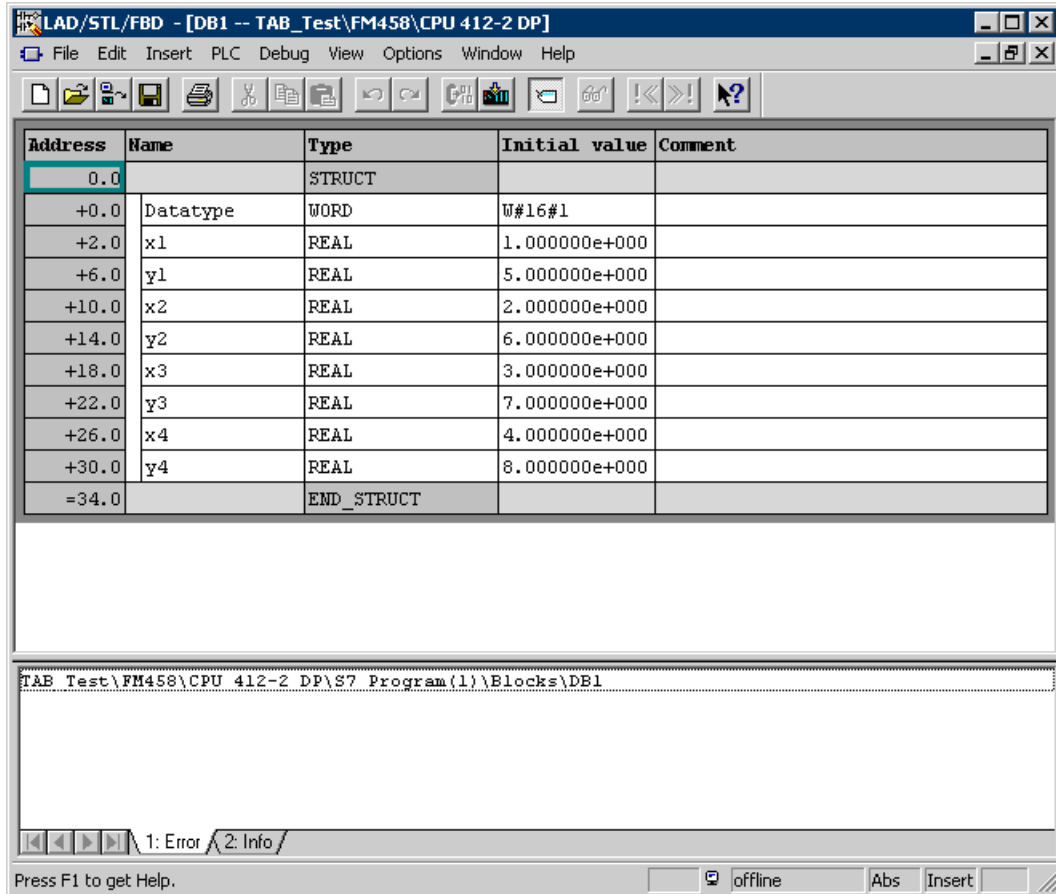


Figure 3-44 Content of the new DB in the "LAD/STL/FBD" application

3.9.3.4 Loading additional tabular values to a DB

If it is necessary to load additional tabular values to the DB because the table is too large and application memory is output space for several DBs, you must transmit the table to the SIMATIC FM 458 application module in several segmented frames. For this purpose, distribute the table to several segments and select a segmentation that prevents overflow of the application memory on the S7 CPU. The table segments will be transmitted successively.

Note

Always make sure that the table segments are transmitted in ascending order of the value pairs. If not transmitted in the proper sequence, the tabular values in the configuration will be corrupted.

Two options are available:

- Enter the table segment manually in the DB in succession using the "LAD/STL/FBD" application and then transmit this table segment
- Generate separate source files with different names for each table segment and then transmit these successively following their successful implementation in the DB

Manual entry

To download additional tabular values to a DB in manual mode, proceed as follows:

- Open the corresponding DB with double-click in the "LAD/STL/FBD" application.
- Replace the existing tabular values with the values of the next table segment
- and save the DB.
- The values of the table segment are now available for transmission.

Generating several source files

To download additional tabular values to a DB by generating several source files, proceed as follows:

- Always specify the same DB name in the header of the source files (*.AWL).
- The size of individual files may not exceed the memory capacity of the DB.
- It is best practice to number file names in ascending order.
- You must now implement the individual files as source files as described above. However, they will not be compiled right away.
- Compile the first source file and then transmit the tabular values that are now available in the DB.
- Compile the second source file to make its tabular values available in the DB. The values are now transmitted to the S7 controller.
- The remaining source files will be compiled and transmitted in succession.
- After the last table segment has been transmitted, reset the LASTDB connection from 0 to 1 to signal the end of transmission.

3.9.3.5 Structure of the data frame for TCP/IP or DUST1 connections

You must observe the structure of the data frames if the communication link is based on TCP/IP or DUST1. This is described in the following. The data frames are "created" using the CTV and CRV function blocks.

The data frame definition allows the transmission of tabular values in single or multiple data blocks.

The following table shows the data block structure:

Data type	Description
char [4]	Frame ID Identifies each table frame with "TAB0" ID
u_int16	Frame commands (bit coded) 1: New table (positive edge from 0 -> 1) 2: Table end
u_int16	Data format (REAL=1, DINT=2)
u_int32	No. of the current data block
u_int32	Number of tabular values (X and Y values) The number of values must always be even, which means that the number of X values equals the number of Y values to be transmitted.
u_int32 [56] / float [56]	Array of tabular values. (X and Y values, always alternating)

The TAB or the TAB_D transmits an acknowledgment frame to the transmitter for each data block received.

The following table shows the structure of the acknowledge frame:

Data type	Description
char [4]	Frame ID Identifies each table frame with "TAB0" ID
u_int32	No. of the current data block
u_int32	Status / error numbers 0xB210OK (data block is OK)

Note

New table data is only transferred to the inactive table if the "New table" command is set.

After the "End of table" command has been received, all additional table data is rejected until the "New table" command is received.

3.9.4 Automatic mode: Memory card

Table values can be combined to form components using the D7-SYS additionalComponentBuilder (included in D7-SYS V5.2 plus SP1), which can be included as additional objects when data is loaded to the memory card. These objects can be read from the memory card with the help of function block TAB, or TAB_D.

One or several table files are imported to the D7-SYS additionalComponentBuilder that combines these files to form a component file (download file) that you can load to the memory card.

The D7-SYS additionalComponentBuilder (aCB) never checks the file content. Tables are an exception to this rule. The content of these table files is checked. The aCB immediately reports corruption of the table file structure.

The example used in the following sections shows the procedures starting with the creation of a table file and ending with the configuration of the function blocks.

3.9.4.1 Creating a table file in csv format

The table values are created as required using a spreadsheet program (e.g. Excel).

The figure shows two Microsoft Excel spreadsheets side-by-side. The left spreadsheet, titled 'Table1.xls', has a table with 18 rows and 5 columns (A-E). Column A contains values from 1.00 to 2.50 in increments of 0.10. Column B contains values from 1.00 to 6.25 in increments of 0.15. The right spreadsheet, titled 'Table2.xls', has a table with 18 rows and 5 columns (A-E). Column A contains values from -1 to 0.5 in increments of 0.1, with a scientific notation value at row 11. Column B contains values from 1 to 0.25 in increments of 0.07, with a scientific notation value at row 11.

Row	A	B
1	1.00	1.00
2	1.10	1.21
3	1.20	1.44
4	1.30	1.69
5	1.40	1.96
6	1.50	2.25
7	1.60	2.56
8	1.70	2.89
9	1.80	3.24
10	1.90	3.61
11	2.00	4.00
12	2.10	4.41
13	2.20	4.84
14	2.30	5.29
15	2.40	5.76
16	2.50	6.25

Row	A	B
1	-1	1
2	-0.9	0.81
3	-0.8	0.64
4	-0.7	0.49
5	-0.6	0.36
6	-0.5	0.25
7	-0.4	0.16
8	-0.3	0.09
9	-0.2	0.04
10	-0.1	0.01
11	-1.38778E-16	1.92593E-32
12	0.1	0.01
13	0.2	0.04
14	0.3	0.09
15	0.4	0.16
16	0.5	0.25

Figure 3-45 Tabular values in Excel

Conditions

The spreadsheet files must fulfill the following conditions:

- A spreadsheet file may only consist of two columns; an error message dialog is output if the table contains additional columns.
- Both columns must contain the same number of values. Otherwise, D7-SYS additionalComponentBuilder outputs an error message dialog and the tabular values are rejected.

The D7-SYS additionalComponentBuilder expects the following data format:

- [+/-] xxx,yyy – REAL value, decimal places are specified in dot notation "." (e.g. 145.123)
- [+/-] xxx,yyy – REAL value, decimal places are specified in dot notation "." (e.g. 145.122)
- [+/-] xxx,yyyE+/-mm – REAL values as exponential, with decimal places specified in dot notation "." (e.g. 145.122E+12)
- [+/-] xxx,yyyE+/-mm – REAL values as exponential, with decimal places specified in comma notation "," (e.g. 187,122E+12)

For the "Table DINT" type definition:

- [+/-]xxx – Integer ,or double integer (e.g. 145)

The following conditions are also valid for the spreadsheet files:

- ASCII files
- Separation of table columns by semicolon or tab character
- Separation of rows by line break or semicolon

Saving tables

Tables created in MS Excel and saved in *.csv format or as "Text (tab separated)" fulfill these conditions.

The following figure shows two sample files with tabular values saved in csv format:

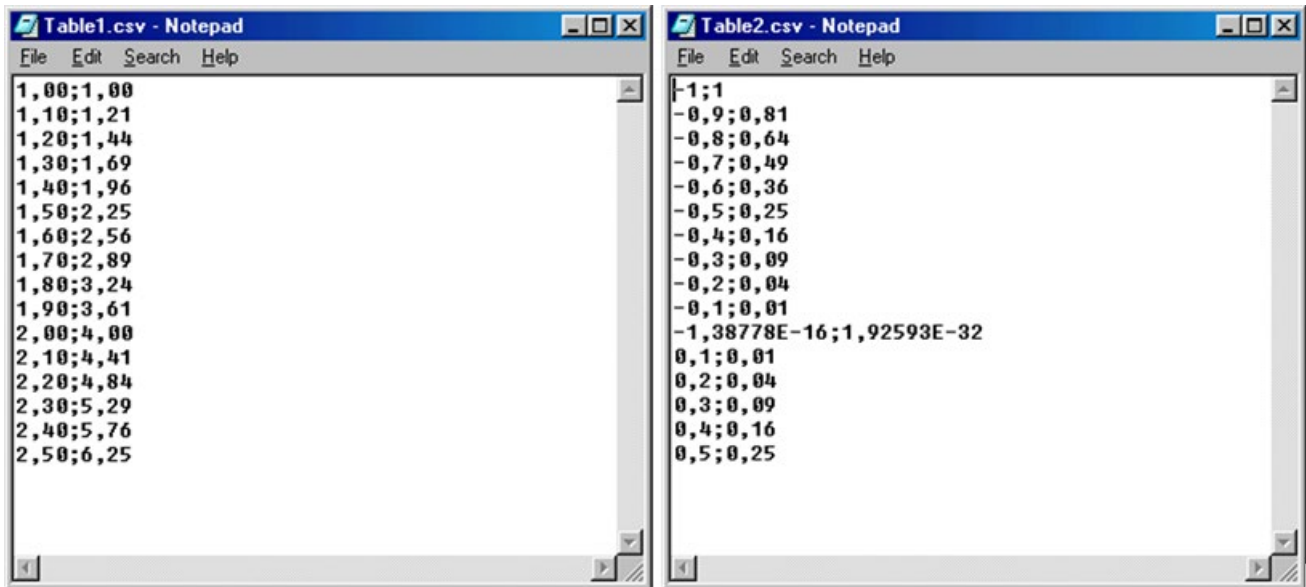


Figure 3-46 Tabular values with semicolon separation (*.csv format)

3.9.4.2 Working with the D7-SYS additionalComponentBuilder

You can import the spreadsheet files that were saved in csv format to the D7-SYS additionalComponentBuilder.

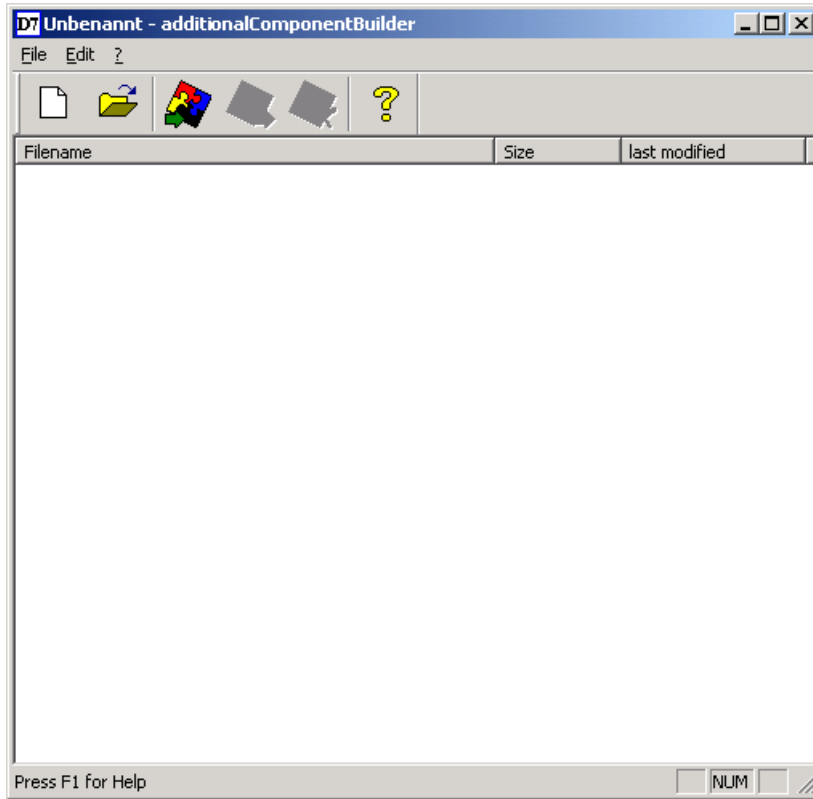


Figure 3-47 D7-SYS additionalComponentBuilder



Create a new component file in the next step. For this purpose, specify the properties in the next dialog.

New component

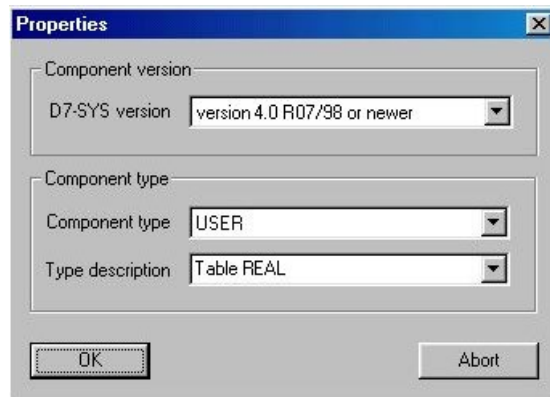


Figure 3-48 Setting the properties

Make the following settings:

Once saved, the property settings are grayed out and cannot be edited.

- **D7-SYS version**

List box that specifies the version for which the component is to be created

- **Component type**

List box with the fixed entries "USER", "IT1", and "IT2". The default value is "USER".

Meaning of the entries:

- USER = Component file created by the user, e.g. spreadsheet files
- IT1/IT2 = System component file for ITSP modules

- **Type definition**

List box with the "Table REAL" and "Table DINT" entries. "Table REAL" is the default value for the "USER" component type. "Table DINT" is used for tables with DINT format.

Meaning of the entries:

- REAL table: Spreadsheet file with data type REAL
- DINT table: Spreadsheet file with Double Integer data type

You can enter a new type definition in the list box and confirm the entry with RETURN. This new type definition is now available in the list box for future selections.

Saving

Once you completed the settings, you can create the new component file.

The new component file is generated in the default path C:\temp. You can specify a different storage path that will be used as default at the restart of the program.

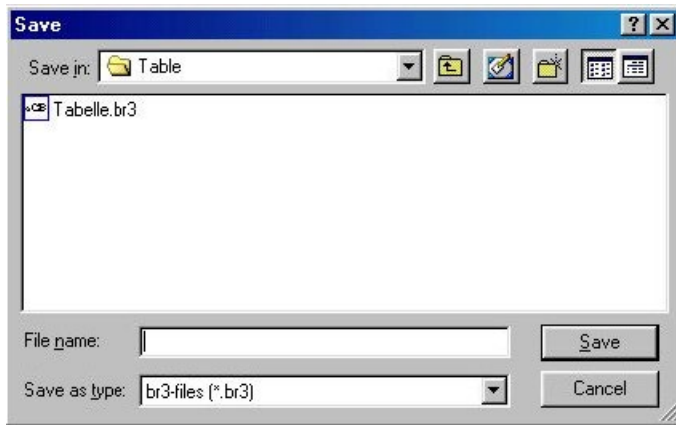


Figure 3-49 Saving the new component file



You can now add spreadsheet files. You can select the spreadsheet files from the file selection window.

Note

A component with type definition "table" must always contain tables with uniform value format! This means that a REAL table only contains tables with REAL values.

The following diagram shows the content of the D7-SYS additionalComponentBuilder after the two spreadsheet file you generated were imported:

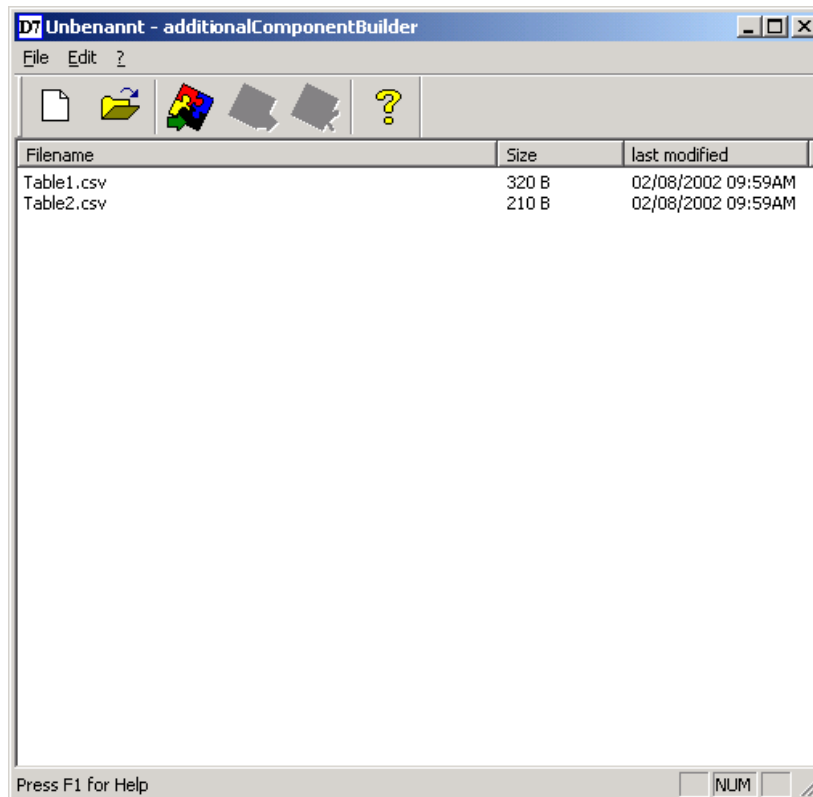


Figure 3-50 D7-SYS additionalComponentBuilder with imported spreadsheet files

You may always add, import, or delete spreadsheet files. D7-SYS additionalComponentBuilder automatically handles spreadsheet file management and saves the modified component files.

Opening

When you open existing components, "C:\temp" is the default search path of the D7-SYS additionalComponentBuilder. If a different search path is selected, this will be set as default at the next restart of the program.

3.9.4.3 Loading

Once the component file has been created in D7-SYS additionalComponentBuilder, you can download the file using the standard download dialog.

(1) Opening the load dialog in D7-SYS with "PLC > Download"

This dialog can be used in the current configuration to load the optional components to a memory card (offline/online mode).

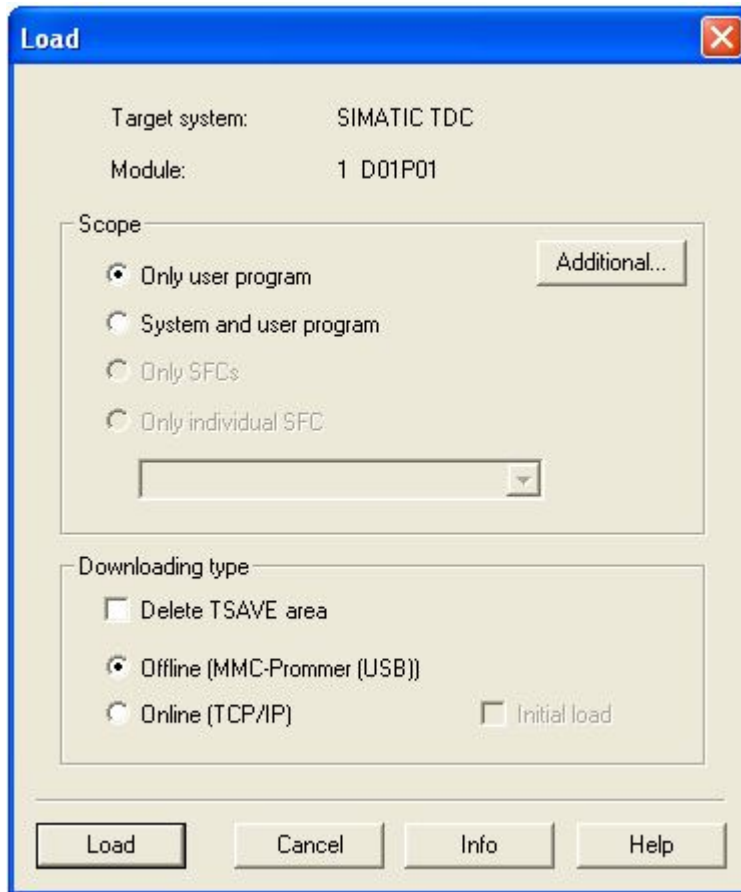


Figure 3-51 Download dialog opened by means of PLC > Download to D7-SYS

(2) Opening the dialog for optional components

You can select up to two components. Click "NEW" to select a file for the selected component.

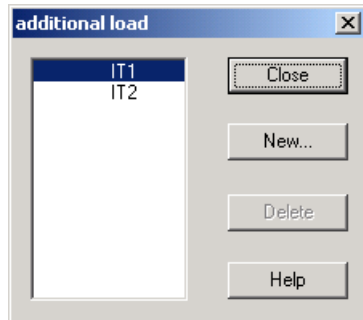


Figure 3-52 Selection dialog for optional components, e.g. spreadsheet data

(3) A file selection dialog opens for selecting additional components

The component file previously created in D7-SYS additionalComponentBuilder is now assigned to component IT1 and written to the memory card in the next load process.

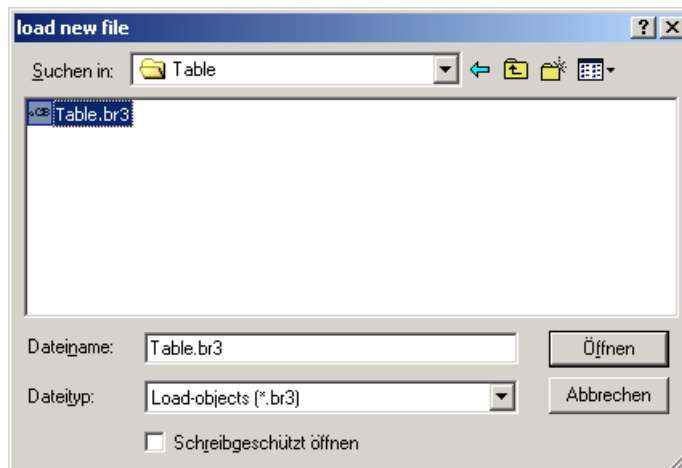


Figure 3-53 Loading component files

3.9.4.4 Configuring the function blocks

You only need to configure the TAB and/or TAB_D function blocks the "Automatic mode:Memory card" mode, depending on whether tabular values of data type REAL and/or DINT have to be managed. Each table may only contain values of the same data type. If you want to manage several tables of different data types, configure a TAB or TAB_D FB for each table.

The TAB and TAB_D function blocks should be configured with a sampling time greater than or equal to 32 ms. The following connection settings must be made:

- CTS = 0
- US = n.c.
- NAM = Name of the spreadsheet file (with file name extension as defined when "saving", e.g. MS Excel)
- AUT = 1 (automatic mode activated)

The following figure shows the configuration:

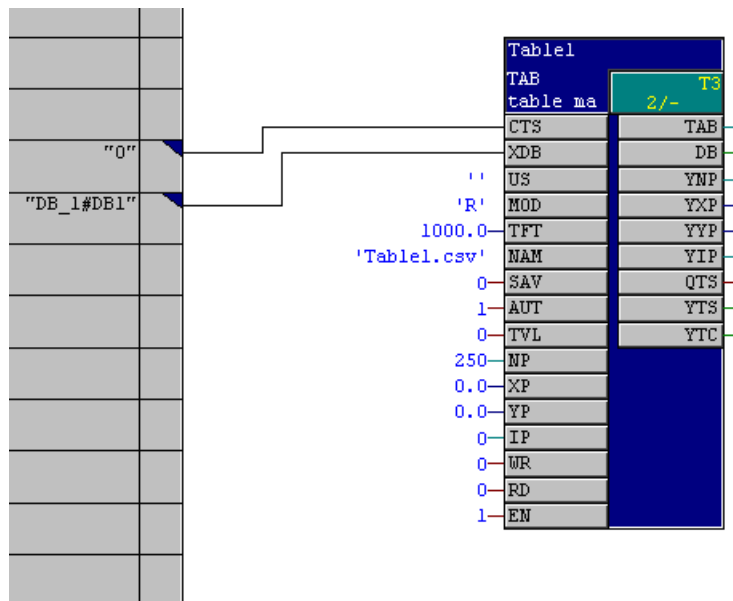


Figure 3-54 Configuration of the TAB function block

The following figure shows the spreadsheet function blocks for two tables. The tabular values now managed by the function blocks are available for use in additional function blocks, e.g. FB TABCAM.

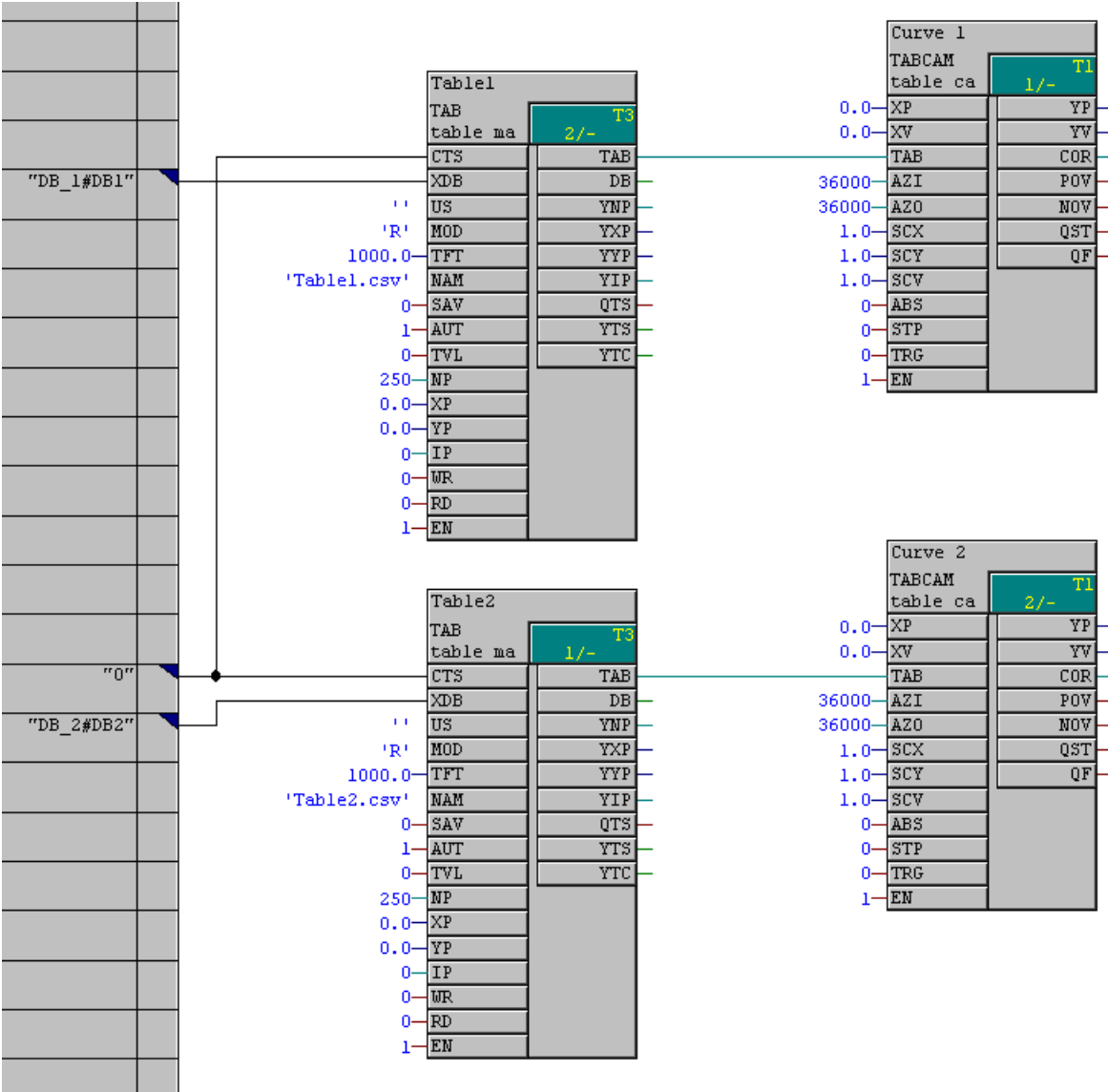


Figure 3-55 Configuration sample

3.10 Communication utility alarm logging

General

Alarm logging allows you to log explicitly selected events. A description of these events is collected in the message sequence buffer and made available to you via data interface.

Configuration

Alarm logging is always operated as local application on the CPU. Precisely one central block and at least one message evaluation block must be configured. There are no configuring rules regarding the number of blocks.

Function blocks for alarm logging

Alarm logging consists of three types of function blocks:

- **Central block @MSC**

The central block creates and manages the necessary data structures. It is also responsible for evaluating communication and system error messages.

- **Alarm logging blocks MER ...**

Alarm logging blocks generate messages as a result of input changes. Alarm logging blocks can interrupt each other. For this reason, the alarms do not have to be entered in the alarm sequence buffer in the order of their occurrence.

The alarm logging blocks differ by:

- the number of alarms which can be generated.
- the options for processing additional incoming process states in the form of measured values.

- **Alarm evaluation block MSI ...**

Alarm evaluation blocks output alarms that are generated by the alarm logging blocks via data interface and make these available to users.

3.10.1 Logging logic of the alarm logging blocks

Alarm blocks for an incoming message

The following logging conditions must be met for alarm logging blocks that only generate incoming alarms:

- Input EN must be set.
- A positive edge must be set at input I1.
- Connection Q1 or SM must be reset.

If all conditions are met, an alarm will be generated and I/O Q1 is set.

If conditions are not met, I/O Q1 is reset if I/O SM has been reset.

Alarm block for an incoming and outgoing alarm

The following logging conditions must be met for alarm logging blocks that generate incoming and outgoing alarms:

- Input EN must be set.
- A positive edge must be set at input I1 and I/O Q1 or SM must be reset for incoming alarms.
- A negative edge must be set at input I1 and I/O Q2 or SM must be reset for outgoing alarms.

If conditions are met:

- An incoming alarm is generated and I/O Q1 is set at the positive edge
- An outgoing alarm is generated and I/O Q2 is set at the negative edge.

If conditions are not met, I/Os Q1 and Q2 are reset if I/O SM has been reset.

Special features for MER16, MERF16, MER0, MERF0

For the alarm logging blocks MER16, MERF16, MER0, and MERF0 that feature a vector as alarm I/O and generate 16 or 32 alarms, corresponding bit positions must meet the logging logic conditions at alarm I/O IS1 and output QS1 or QS2.

A QN output at these blocks indicates whether or not an alarm was generated.

3.10.2 Example of an alarm logging configuration

Alarm logging requirements

- Rack
- The rack contains at least one CPU
- A data interface named "D01" is available

Necessary function blocks

The example only lists the blocks actually required for alarm logging. Central communication blocks (e.g. for the data interface) are not listed.

The configured alarm logging consists of:

- One central block @MSC
- Two alarm input blocks (MER and MERF0)
- Two alarm evaluation blocks (MSI and MSIPRI)

Name and alarm buffer

Alarm logging is assigned the name "MELD". This name is configured for all CMS connections of the alarm blocks. The alarm buffer has capacity for 30 alarms (connection NOM at @MSC), is located in volatile RAM (connection SAV at @MSC) and enabled for alarm entries (connection MUN at @MSC).

Assigning alarms and blocks

Alarms generated can be assigned to blocks by means of RP and RRS connections, with every block of alarm logging having at least one RP connection. Proceed as follows:

- **Prefix 0**
Identifies alarms generated by @MSC (communication and system error messages). For this reason, connection RP of @MSC is assigned the value 0. @MSC automatically generates the suffix, depending on the alarm type.
- **Prefix 1**
Identifies alarms generated by MSI (overflow alarms). For this reason, connection RP if MSI is assigned the value 1. MSI automatically generates the suffix (number of alarms having overflowed).
- **Prefix 2**
Identifies alarms generated by MSIPRI.

- **Prefix 3**

Identifies alarms generated by an alarm block (MER or MERF0). For this reason, connection RP of MER and MERF0 is assigned the value 3. In contrast to other blocks, the suffix is not generated automatically and certain connections are available for configuring the suffix. In the example, 33 different alarms are generated and numbered from 0 - 32 (1 MER alarm, 16 incoming alarms MERF0, 16 outgoing MERF0):

- The alarm of block MER is assigned suffix 0 (RS connection MER).
- The 16 incoming alarms of block MERF0 are assigned suffix 1-16 (RS1 connection MERF0).
- The 16 outgoing alarms of block MERF0 are assigned suffix 17-32 (RS2 connection MERF0).

- **Suffix**

At the suffix for the MERF0 block, a basic value is specified to which the bit number of the alarm-generating bits of alarm signal vector IS1 is added.

Functional grouping of the alarms

The prefix and suffix enable unambiguous alarm assignment to the generating blocks, as well as the functional grouping of alarms. In the sample configuration, the MER and MERF0 blocks generate alarms having the same prefix that indicates a logical relation.

Channel on the data interface

In the sample configuration, both alarm evaluation blocks set up a channel at data interface D01 in the "Select" mode (which means that you could assign the same channel name).

Measured value input and alarm signals

The measured value input of block MER is not wired in the this example. Usually, a process status is set at the measured value input. The same applies to the alarm signals of function block MERF0.

Generating and reading alarms

Alarms are generated by a positive edge at connection I1 of block MER, or by changes to the value at input IS1 of block MERF0. The alarm evaluation block would immediately read the first alarm from the alarm buffer and transfer it to the data channel, as both blocks are "enabled" (input EN=1). Additional alarms are only transferred to the data channel after the previous alarm has been read from the channel.

3.10 Communication utility alarm logging

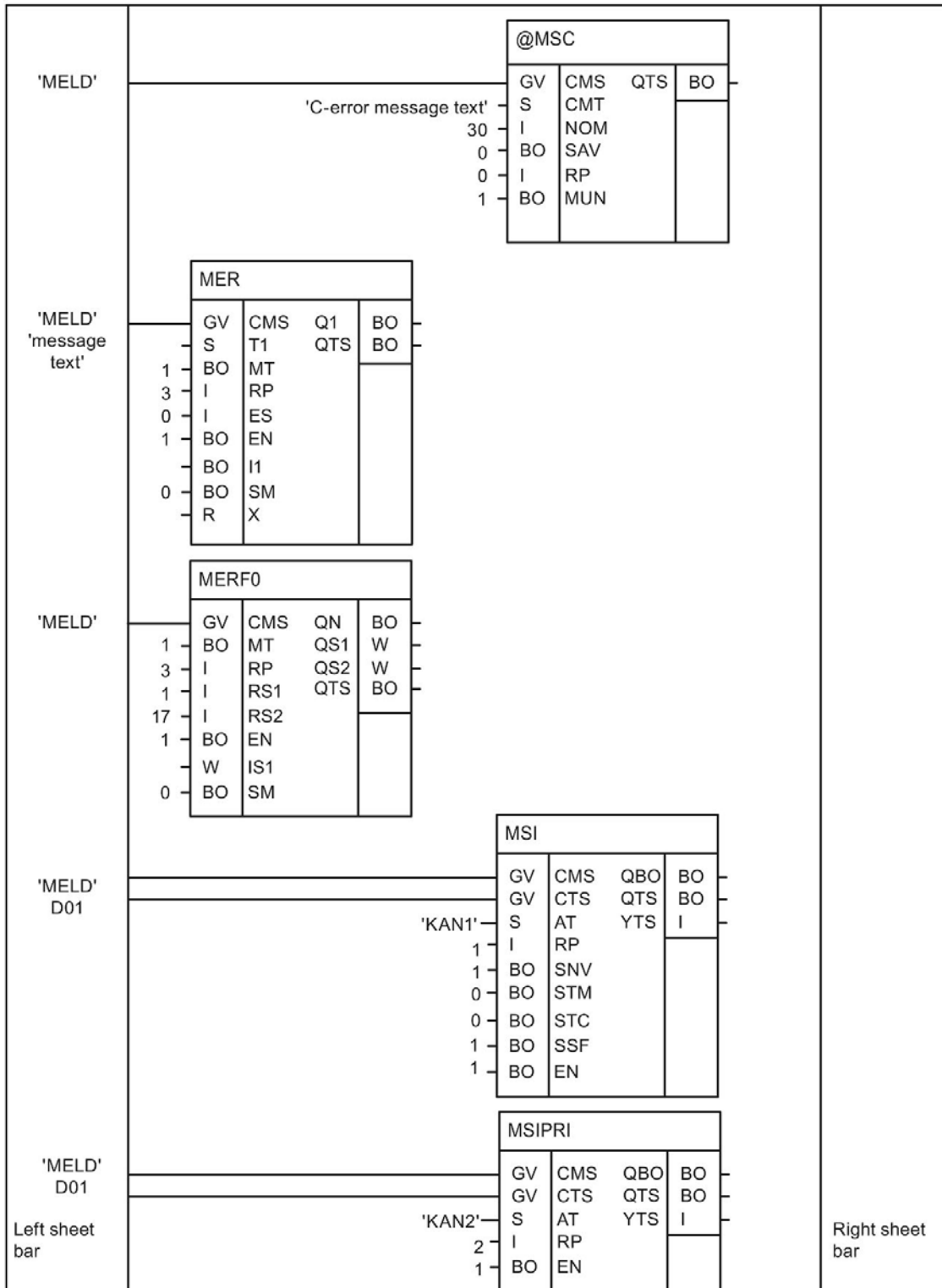


Figure 3-56 Example of the communication utility alarm logging

3.10.3 Output formats of the alarm evaluation block MSI

3.10.3.1 Structure of an error or alarm message

General

The alarm evaluation block MSI provides four connections for selecting the format:

- Connection SNV
- Connection STM
- Connection STC
- Connection SSF

The alarm format is important for the receiver of an alarm and its interpretation.

Length of the alarm text

Connection STC defines the length of the alarm text. If STC = 1, the text is set to a constant length (60 characters). An alarm text that is shorter than the maximum length or not available is padded with white spaces. The advantage lies in the constant amount of data to be transferred as demanded by certain communication partners. This connection has no effect on the remaining structure and type definition of the alarms.

Alarm text format

The SNV, STM and SSF connections are evaluated once in the initialization phase and then define the format of alarm output. They are output at the channel that is specified at connection AT on the data interface specified at the CTS connection.

3.10.3.2 Overview of alarm formats

Spontaneous identification

The spontaneous ID is assigned the constant value 0 and is of no significance.

Sequence number

The sequence number is provided for reasons of reliability and counts the number of alarms transmitted so that the receiver can identify which alarms have been lost. The sequence is in the range of values from 0 to 255. Once the number has reached its maximum value of 255, the minimum value 0 will be transmitted for the next alarm.

Alarm type definition

We basically distinguish between the standardized and HEX formats. With standardized format, the individual values are transmitted in accordance with IEEE 754 or ISO 646 that defines a standardized 32-bit floating point notation. The alarms, both in the standardized and HEX formats, include an alarm type definition that provides information on the format selected by the initialization connections and other alarm components. The alarm type definition is a bit vector that must be interpreted as follows:

- Bit 0: Alarm numbers will be output (copy of connection SNV) if this bit is set.
- Bit 1: An alarm text is output (copy of connection STM, unless the alarm text is empty) if this bit is set.
- Bit 2: Alarms are output in standardized format and otherwise in HEX format (copy of connection SSF) if this bit is set.
- Bit 3: A measured value is available if this bit is set.
- Bit 4: A unit text is available if this bit is set. The units text cannot be available unless a measured value is also available. If there is no measured value or unit text, the corresponding alarm fields are of no significance and in undefined state.
- Bits 5-7: n.c.

Alarm type

The alarm type consists of a character that defines the alarm event type, whereby "S" identifies system errors, "C" communication errors, "F" error messages, and "W" alarms. The first two alarm types are only generated by the central block of alarm logging.

Alarm prefix

Corresponds to the value set at connection RP of the input block.

Alarm suffix

Corresponds to the value at connection RS of the input block.

Measured value units and scaling factor

A measured value definition in HEX format consists of

- a 32-bit scaling factor that is output in floating format
- the measured value logged by the input block
- a measured value data type (SIMADYN D / SIMATIC TDC data type as ASCII character string)
- a measured value unit with a length of 8 characters

HEX format and standardized format

As the precise data format must be specified in HEX format for initialization of the transmission channel and the length of the measured value displayed may vary (0, 2, or 4 bytes), 4 bytes are always transmitted for measured values. If the measured value occupies less than 4 bytes, which can be recognized at the measured value data type, the remaining bytes will not be used.

In standardized format, only the scaled measured value and the measured value unit with a length of 8 characters will be transmitted.

Alarm time

The alarm time is transmitted in HEX notation in the MMS format, Time and Date (reference point 1.1.84).

In standardized format, the alarm time is transmitted as ASCII string that includes the date (day, month, year) and time (hours, minutes, seconds, milliseconds). The date and time are separated by a white space. The string has a length of 24 characters (example: "01.05.1993 08:01:15:0045").

Alarm text

The alarm text is always transmitted as ASCII string and does not include length information. This length must be calculated based on the total amount of data received. The alarm text may not exceed a length of 60 characters.

3.10.3.3 Structure of an overflow alarm

Overflow alarm

On overflow of the alarm sequence buffer, the MSI/MSPRE generates an overflow alarm:

- The overflow alarm is of the warning type ('W').
- The prefix includes the value at connection RP of function block MSI that generates the alarm.
- The suffix includes the number of alarms that have been lost.
- There is no measured value. This is indicated in the alarm type definition.
- The alarm event time is entered as the time at which the alarm evaluation block generated the overflow alarm.
- The "sequence buffer overflow" text is output as alarm text if the STM connection of function block MSI is set.

3.10.3.4 Structure of a communication error message

Communications error message

The central block evaluates the communication errors that occur in the system and generates the following communication error messages:

- A communications error message is an alarm type C error ('C').
- The prefix includes the value at the RP connection of the central block that generated the alarm.
- The suffix includes the error number (always positive) of the C error message.
- Unavailable measured values are indicated in the alarm type definition.
- The text configured at connection CMT of the central block is output as alarm text if connection STM of function block MSI is set.
- On overflow of the communication error field and after all C error messages have been output, an alarm will be generated that contains the negative number of lost alarms as suffix. MSI does not output any additional C error messages after this alarm. The instant at which the central block detected the communication error field overflow is entered as the alarm event time.

3.10.3.5 Structure of a system alarm

System alarm

The structure of a system alarm is basically the same as for communication error messages, with differences with regard to the alarm text for which the "system alarm" is always used, as well as the alarm type ('S'). Moreover, a maximum of one system alarm will be generated, which is identified in the initialization phase of the central block.

An ID is entered for the system alarm as suffix that has the following significance:

Table 3- 10 System alarm suffix

Suffix value	Significance
1	5 V power failure
2	15 V power failure
3	Malfunction in software runtime
4	Error when accessing the L bus buffer memory
5	Error when accessing the C bus buffer memory
6	Error when accessing standard I/O devices
7	Error when accessing the special I/O devices
8	Undefined L bus access
9	Undefined C bus access
10	(n.c.)
11	Hardware fault which cannot be identified
12	(n.c.)
13	Error which cannot be identified
14	Error message (Ready internal) from local expansion bus (LE bus)
15	Error when accessing local peripherals (LP bus)
16	Overrun of the system bus controller

3.10.3.6 Detailed description of the alarm formats of function block MSI

General

The alarm format definition consists of three elements:

- Pin assignment of the initialization connections SNV, STM, and SSF
- Basic format and maximum length of the alarm. This length corresponds to the size of the channel registered by MSI.
- User data structure required for channel initialization.
- Connection STC is not included in this list. If STC = 1, the length specification for the alarm text always corresponds with the maximum length; if STC = 0, it corresponds to the actual alarm text length.

Table 3- 11 Standard format with number and text

SNV=TRUE (alarm numbers available) STM=TRUE (alarm text available) SSF=TRUE (standardized format)				
Content	Alarm structure (max. 108 bytes)	User data structure	Data format	Amount of data
Spontaneous identification	Unsigned8	1 variable unit	Unsigned8	2
Sequence number	Unsigned8			
Alarm type definition	1 Octet	2. variable unit	Octet String	2
Alarm type	1 Octet			
Prefix	Floating point	3. variable unit	Floating point	3
Suffix	Floating point			
Measured value	Floating point			
Measured value dimensions text	8 characters	4. variable unit	Visible string	92
Alarm time	24 characters			
Alarm text	max. 60 characters			

Table 3- 12 Standard format without number with text

SNV=FALSE (alarm numbers unavailable)				
STM=TRUE (alarm text available)				
SSF=TRUE (standardized format)				
Content	Alarm structure (max. 100 bytes)	User data structure	Data format	Amount of data
Spontaneous identification	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Alarm type definition	1 Octet	2. variable unit	Octet String	2
Alarm type	1 Octet			
Measured value	Floating point	3. variable unit	Floating point	1
Measured value dimensions text	8 characters	4. variable unit	Visible string	92
Alarm time	24 characters			
Alarm text	max. 60 characters			

Table 3- 13 Standard format with number without text

SNV=TRUE (alarm numbers available)				
STM=FALSE (alarm text unavailable)				
SSF=TRUE (standardized format)				
Content	Alarm structure (max. 48 bytes)	User data structure	Data format	Amount of data
Spontaneous identification	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Alarm type definition	1 Octet	2. variable unit	Octet String	2
Alarm type	1 Octet			
Prefix	Floating point	3. variable unit	Floating point	3
Suffix	Floating point			
Measured value	Floating point			
Measured value dimensions text	8 characters	4. variable unit	Visible string	32
Alarm time	24 characters			

3.10 Communication utility alarm logging

Table 3- 14 Standard format without number and text

SNV=FALSE (alarm numbers unavailable) STM=FALSE (alarm text unavailable) SSF=TRUE (standardized format)				
Content	Alarm structure (max. 48 bytes)	User data structure	Data format	Amount of data
Spontaneous identification	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Alarm type definition	1 Octet	2. variable unit	Octet String	2
Alarm type	1 Octet			
Measured value	Floating point	3. variable unit	Floating point	1
Measured value dimensions text	8 characters	4. variable unit	Visible string	32
Alarm time	24 characters			

Table 3- 15 HEX format with number and text

SNV=TRUE (alarm numbers available) STM=TRUE (alarm text available) SSF=FALSE (HEX format)				
Content	Alarm structure (max. 92 bytes)	User data structure	Data format	Amount of data
Spontaneous identification	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Alarm type definition	1 Octet	2. variable unit	Octet String	2
Alarm type	1 Octet			
Prefix	Unsigned16	3. variable unit	Unsigned16	2
Suffix	Unsigned16			
Measured value scaling factor	Floating point	4. variable unit	Floating point	1
Measured value	4 Octets	5. variable unit	Octet String	6
Measured value data type	2 Octets			
Measured value dimensions text	8 characters	6. variable unit	Visible string	8
Alarm time	Time and Date	7. variable unit	Time and Date	1
Alarm text	max. 60 characters	8. variable unit	Visible string	60

Table 3- 16 HEX format without number with text

SNV=FALSE (alarm numbers unavailable)				
STM=TRUE (alarm text available)				
SSF=FALSE (HEX format)				
Content	Alarm structure (max. 88 bytes)	User data structure	Data format	Amount of data
Spontaneous identification	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Alarm type definition	1 Octet	2. variable unit	Octet String	2
Alarm type	1 Octet			
Measured value scaling factor	Floating point	3. variable unit	Floating point	1
Measured value	4 Octets	4. variable unit	Octet String	6
Measured value data type	2 Octets			
Measured value dimensions text	8 characters	5. variable unit	Visible string	8
Alarm time	Time and Date	6. variable unit	Time and Date	1
Alarm text	max. 60 characters	7. variable unit	Visible string	60

Table 3- 17 HEX format with number without text

SNV=TRUE (alarm numbers available)				
STM=FALSE (alarm text unavailable)				
SSF=FALSE (HEX format)				
Content	Alarm structure (max. 32 bytes)	User data structure	Data format	Amount of data
Spontaneous identification	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Alarm type definition	1 Octet	2. variable unit	Octet String	2
Alarm type	1 Octet			
Prefix	Unsigned16	3. variable unit	Unsigned16	2
Suffix	Unsigned16			
Measured value scaling factor	Floating point	4. variable unit	Floating point	1
Measured value	4 Octets	5. variable unit	Octet String	6
Measured value data type	2 Octets			
Measured value dimensions text	8 characters	6. variable unit	Visible string	8
Alarm time	Time and Date	7. variable unit	Time and Date	1

Table 3- 18 HEX format without number and text

SNV=FALSE (alarm numbers unavailable) STM=FALSE (alarm text unavailable) SSF=FALSE (HEX format)				
Content	Alarm structure (max. 28 bytes)	User data structure	Data format	Amount of data
Spontaneous identification	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Alarm type definition	1 Octet	2. variable unit	Octet String	2
Alarm type	1 Octet			
Measured value scaling factor	Floating point	3. variable unit	Floating point	1
Measured value	4 Octets	4. variable unit	Octet String	6
Measured value data type	2 Octets			
Measured value dimensions text	8 characters	5. variable unit	Visible string	8
Alarm time	Time and Date	6. variable unit	Time and Date	1

3.10.3.7 Output format of the message evaluation block MSIPRI

General

In contrast to alarm evaluation block MSI, you can select the format of the alarms of evaluation block MSIPRI. Here, only one format is output. The connections for select a format are discarded accordingly when you configure the block. The MSIPRI block was developed with the focus set on alarm output to a printer. All alarms are output in text form and formatted with line feed. An alarm consists of up to two lines.

Structure of the first line

Table 3- 19 Structure of the first line in the alarm of evaluation block MSIPRI

Characters in the first line	Significance	Output format
1-24	Date/time	Day.Month.Year, Hour:Minute:Second:Millisecond
25-27	Text: "P:"	
28-32	Prefix	Max. 5 characters and right justified
33-35	Text: "S:"	
36-40	Suffix	Max. 5 characters and right justified
41-45	Text: "Type:"	
46	Alarm type ('C','F','W', or 'S')	One character
47-50	Text: "No.:"	
51-53	Sequence number	Max. 3 characters and right justified
54	Text: " "	
55-67	Measured value (optional: only entered if the alarm contains a measured value)	Output as floating value in the following order: <ul style="list-style-type: none"> • Sign (positive = "+", negative = "-") • Number of digits leading the decimal point, followed by the decimal point and 6 decimal places • Exponent, started with the character 'e' • Sign (positive = "+", negative = "-") as well as two exponent digits
68	White spaces (optional)	
69-76	Measured value unit (optional: only entered if the alarm contains a measured value)	8 characters
77, 78	Special characters CR and LF	Line feed

Structure of the second line

The second line contains the alarm text; only output if an alarm text is available and discarded otherwise.

Table 3- 20 Structure of the second line in the alarm of evaluation block MSIPRI

Characters in the second line	Significance	Output format
1-60	Measured value text (optional)	Variable length
61, 62	Special characters CR and LF	Line feed

Example of a message output

"01.05.1993 08:01:15:0045 P: 123 S: 10 Type: W No: 25 -1.123456e+12 ms"

"This is an alarm text"

Note

Overflow, communication error, and system alarms have the same logical structure as the MSI block.

3.11 Communication utility process data

Application

Communication utility process data supports "pure" bidirectional data transmission, i.e. the function blocks only transmit process data. The data itself is not evaluated or logically interpreted.

There are two block classes for data transmission:

- Receive and transmit blocks: CRV and CTV
- Channel marshalling blocks: CCC4 and CDC4

The CRV and CTV blocks can handle most of the communication applications.

3.11.1 Receive and transmit blocks

General

There is one receive and one transmit block, which are named CRV (communication receive virtual) and CTV (communication transmit virtual).

A receive or transmit block is used to configure a frame that is transmitted from or to a coupling module. The frame structure and content are defined based on the configuration of virtual connections.

3.11.1.1 Virtual connections

General

A virtual connection is an "invisible" connection between block I/O. No connector is drawn on the configuration interface and only a sheet bar connection is created instead.

The configuration engineer specifies the values to be transmitted from block outputs or to block inputs based on the "receive/transmit connection definition" at the receivers or transmitters, as well as on the "virtual connection definition" and "sequence number" at the block I/O to be processed.

Connection definition

The connection definition consists of an exclamation mark ("!") and up to 6 characters (uppercase or numbers). The string is appended directly to the exclamation mark (e.g. "!SEND"). It is not necessary to configure the exclamation mark, as this is generated automatically.

A virtual connection consists of the following elements:

- Virtual connection definition
- Sequence number

The connection definition and sequence number are dot separated (e.g. "!SEND.0056"; it is not necessary to configure the dot separating the connection definition and sequence number, as this is inserted automatically).

Data types

Virtual connections can be configured at I/O with the following data types:

- BOOL (BO), BYTE (BY)
- WORD (W), DOUBLE WORD (DW)
- INTEGER (I), DOUBLE INTEGER (DI)
- REAL (R) and SDTIME (TS)

Note

Virtual connections cannot be configured at I/O of data type STRING (S) or GLOBAL VARIABLE (GV).

Frame structure

Virtual connections having the same connection definition define a frame having a specific structure. The order of the data contained in the frame is determined by the sequence number. The data with the lowest number is entered at the beginning of the frame, while data with the highest number is entered at the end. The sequence number defines the relative position of the parameter in the frame. Gaps in the sequence numbers are ignored.

Sample configuration

Receiving and transmitting with virtual connections.

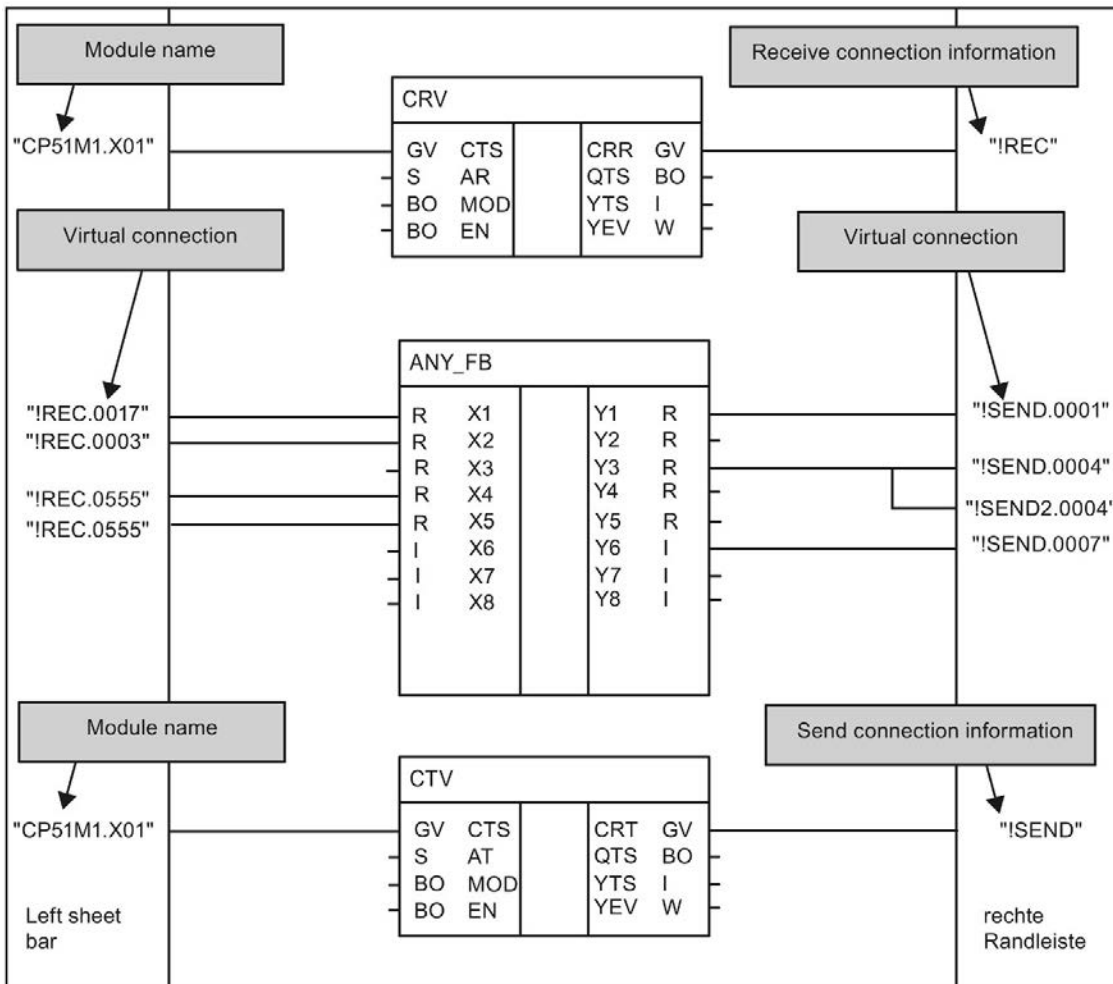


Figure 3-57 Configuration: Receiving and transmitting with virtual connections

Configuration rules with reference to the example:

- Virtual connections that belong to a virtual connection definition can be configured in any order at block I/O having different data types (ANY_FB.Y1 "REAL" and ANY_FB.Y6 "INTEGER").
- You can configure several virtual connections (receive) at block inputs having the same data type. These inputs are provided identical data. (ANY_FB.X4 and X5)
- It is not allowed to configure several instances of the same virtual connection (transmit) with identical connection definition and sequence number at the block outputs.
- However, you may assign several different virtual connections (transmit) to a block output (ANY_FB.Y3). The connections may differ in terms of the connection definition and sequence number.

Frame structure of the connection definition "!REC" from the example:

Table 3- 21 Frame structure of the connection definition "!REC"

Connection	Virtual connection	Data type	Length
ANY_FB.X2	!REC.0003	R	4
ANY_FB.X1	!REC.0017	R	4
ANY_FB.X4/X5	!REC.0555	R	4
			Total length = 12 bytes

Frame structure of the connection definition "!SEND" from the example:

Table 3- 22 Frame structure of the connection definition "!SEND"

Connection	Virtual connection	Data type	Length
ANY_FB.Y1	!SEND.0001	R	4
ANY_FB.Y3	!SEND.0004	R	4
ANY_FB.Y6	!SEND.0007	I	2
			Total length = 10 bytes

The structure of the configured frame appears in similar form in the CFC reference data in the "Cross-references, addresses" view, or in the CPU MAP listing (of the CFC) under the keyword "virtual connections". The configuration can be checked based on these lists.

Note

Note the following:

- The virtual connection definitions are known locally on the CPU. Data from different function charts, but not from different CPUs, can be combined to form a frame.
- The data is processed by the receive/transmit blocks within their sampling time. The sampling times of the blocks containing the virtual connections have no influence on the frame processing cycle.
- The configuration engineer is responsible for ensuring compatibility of the frame structure and length with the coupling partner (Function principle of couplings (Page 95)). These rules are dependent on the sub-level coupling. In an error occurs, the receive/send block is deactivated automatically and enter corresponding information in the communication error field (e.g. PROFIBUS DP/PROFINET or rack coupling), or communication cannot be established (e.g. INDUSTRIAL ETHERNET).

3.11.1.2 Connections of the CRV, CTV blocks

CTS I/O

The CTS I/O of the block returns the name of the configured coupling module to be used for communication. An additional connector specification (X01 with CP51M1) is necessary for the CP51M1 block types. An additional connector specification (e.g. X01 for CP51M1) is necessary for some module types.

Connection AR, AT

The address parameter for communication is specified at connection AR, AT. It consists of a channel name and the optional address levels. The significance of the address parameters depends on the coupling used (e.g. PROFIBUS or PROFINET).

Input MOD

The data transmission mode is configured at the MOD input (e.g. "R" for Refresh or "H" for Handshake)

Input EN

Input EN defines whether or not the data is transmitted in the current operating cycle.

Connection CRR, CRT

The virtual connection definition for receiving or transmission is configured at connection CRR or CRT.

Outputs QTS, QT, YEV, YTS

These outputs return information on the block state as well as any error information.

3.11.2 Channel marshalling blocks CCC4 and CDC4

Application

Channel marshalling blocks are used to split or group channels.

3.11.2.1 Group block CCC4

General

Function block CCC4 (Communication Collect Channel 4) allocates up to four channels to a single channel. The channels may have different address definitions, be located on different data interfaces, and have different data transmission modes or lengths.

Requirements

At least two channels must be to enable operation of the function block (CT1 and CT2 connection definitions are mandatory).

Definition at connections CT3, CT4

If only two channels are to be combined, configure a "0" (zero) value for the CT3 and CT4 initialization connections. In this case, AR3, AR4, MO3, MO4, LT3, and LT4 are no longer evaluated.

Definition at input CTS, AT, MOD

The transmission channel is specified at the CTS, AT, and MOD connections. The length of user data to be transmitted is derived from the total of receive data. Receive channels 1-4 are grouped in succession to form a large user data block.

Example

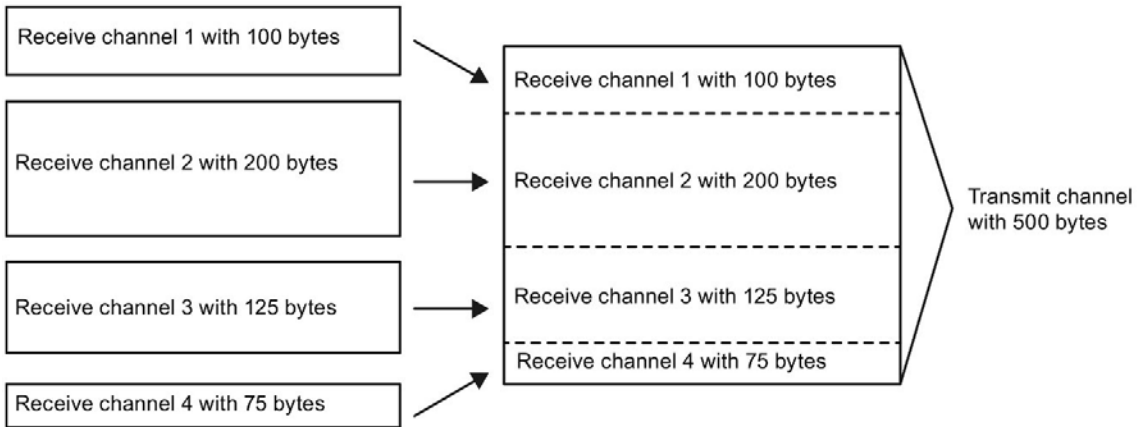


Figure 3-58 Grouping four receive channels to form a transmission channel

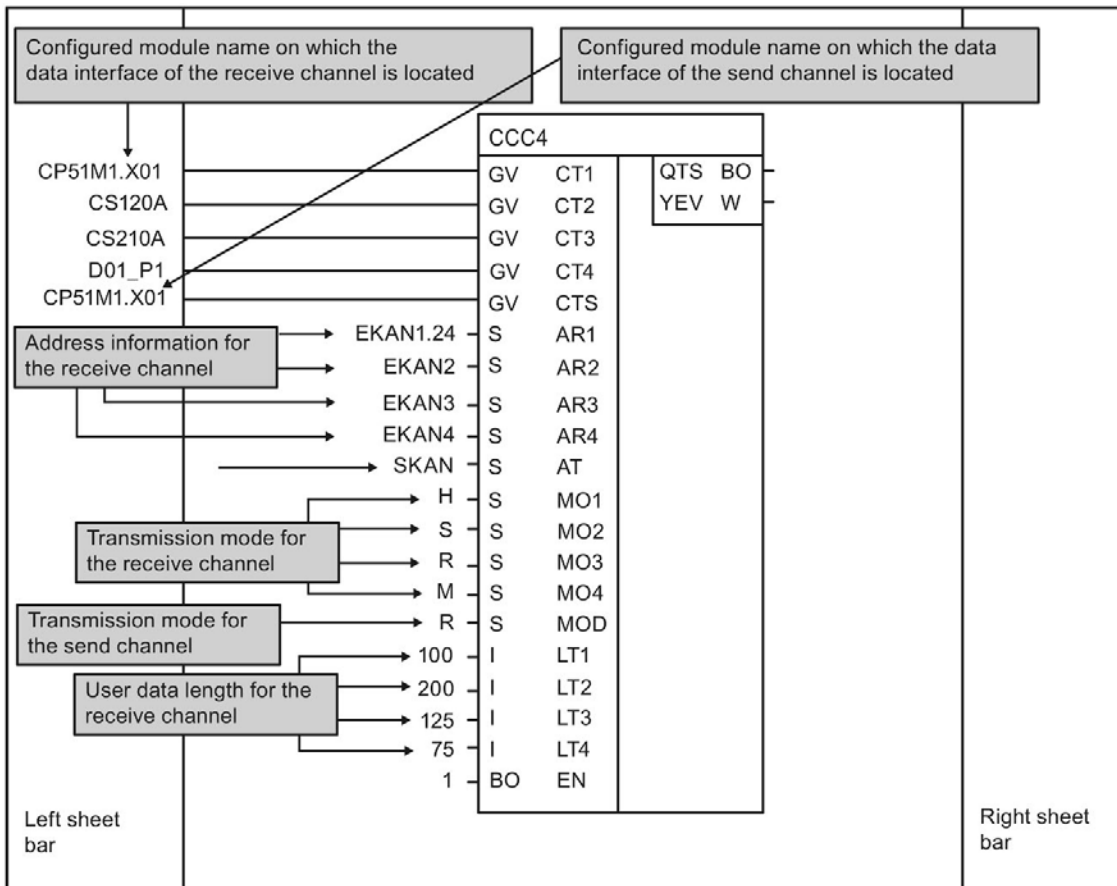


Figure 3-59 Sample configuration CCC4 connections with four grouped channels

3.11.2.2 Distribution block CDC4

General

Function block CDC4 (Communication Distribute Channel 4) splits a channel into up to four channels. The channels may have different address definitions, be located on different data interfaces, and have different data transmission modes or lengths.

Requirements

To enable operation of the function block, the receive channel must be split into at least two transmit channels (CT1 and CT2 connection definitions are mandatory).

Definitions at the CT3, CT4 connections

If a channel is distributed to two channels only, configure a "0" (zero) value for the CT3 and CT4 initialization connections. In this case, AR3, AR4, MO3, MO4, LT3, and LT4 are no longer evaluated.

Definition at the CTS, AT, and MOD connections

The receive channel is specified at the CTS, AT, and MOD connections. The length of user data to be received is derived from the total of transmission data.

Note

If one of the transmission channels is configured for the Handshake mode and this particular channel is not read on receiving side, operation of the CDC4 function block is prevented until this channel has been read. In this case, the block is temporarily inhibited.

3.11.2.3 Compatible user data structure

The user data (data type octet string) is unstructured at the CCC4 and CDC4 blocks, which means that this data is compatible with any user data structure. Only the length of user data must be identical for correct synchronization of the transmitter and corresponding receiver.

3.11.3 Diagnostics outputs

General

After every processing cycle, the result of processing at the data interface(s) is returned at output YEV of the transmit and receive blocks (CTV, CRV and RDREC) and at the channel marshalling blocks (CCC4, CDC4). Output YEV is of the type WORD and its 16 bits are distributed to three areas:

Table 3- 23 Diagnostics outputs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Channel states (only CCC4, CDC4)					Channel assignment (only CCC4, CDC4)			Error cause							

3.11.3.1 Error cause

Hexadecimal value

The possible cause of error is indicated by the bits 0 to 7 in the form of a hexadecimal value (no bit coded evaluation):

Table 3- 24 Error cause

HEX value	Meaning	Remedy
0	No error, data transmission was successful.	
1	The block was permanently deactivated after initialization due to a configuration error or internal error (for precise message information, refer to the communication error field).	Debugging the configuration.
2	Communication partner not ready, or a hardware error has interrupted the communication path	Check the coupling partner, cables, and connectors.
3	Communication partner is not transmitting/receiving (depending on the enable input of the communication partner). The function block is not transmitting/receiving because the communication partner has signaled that it is not transmitting data, too.	Activate the communication partner
4	Only for the transmitter: No data could be transmitted. (Usually in Handshake/Select mode: the communication partner has not yet read the last data; rarely occurring Refresh mode: the communication partner is busy with reading).	Configure a slower transmitter, to a faster receiver.
5	Only for the receiver: No data could be received. (the communication partner has not transmitted any new data since the last data was received).	Configure a slower receiver, or a faster transmitter.
6	Inconsistent data (rack coupling: at the shutdown of the master rack)	None (continued automatically with re-initialization)
7	Only Select transmitters: Channel in use. Another function block is currently transmitting.	Coordinate all Select transmitters via enable input.
8	Only multiple receivers: Reception failure. Data reading took too long; in the meantime, the transmitter has already written new data to the channel.	Configure a receiver with a faster (higher-priority) sampling time.
9	Initialization is busy. For this reason, transmit/receive operation could not be started.	

Comment to numbers 4 and 5

Sporadic occurrence of these numbers in handshake mode is acceptable, as it is not always possible to synchronize the communication partners. Receivers and transmitters should operate approximately in the same cycle.

Occurrence of these numbers in refresh mode must be avoided if the transmitter is generally faster than the receiver.

3.11.3.2 Channel assignment

General

The area is only used by the CCC4 and CDC4 function blocks. A number output in this area indicates the channel that is related to the source of error (bits 0 to 7). As the channel marshalling block can process up to five channels, the numbering is as follows:

Table 3- 25 Channel assignment

Hex Value	Channel
0	Master sender/receiver (according to the CTS, AT, or AR connection definition).
1	Send/receive part 1 (according to the CT1, AT1, or AR1 connection definition)
2	Send/receive part 2
3	Send/receive part 3
4	Send/receive part 4

3.11.3.3 Channel states

General

The area is only used by the CCC4 and CDC4 function blocks. This area indicates the channels that could not be processed without errors.

The "channel states" area specifies the channels at which malfunctions were detected during channel processing.

This area is bit-oriented:

- 1=no fault
- 0=fault

Table 3- 26 Channel states

Bit	Channel
11	Transmit/receive part 1
12	Transmit/receive part 2
13	Transmit/receive part 3
14	Transmit/receive part 4
15	Master transmitters/receivers

3.11.4 Getting started with "pointer-based communication blocks"

Up to D7-SYS V6, serial or parallel data transmission was configured for SIMATIC control systems using the so-called "virtual communication connections" method (visualized in CFCs e.g.: "!VNAME.0001").

Exception: The fiber-optic drive coupling SIMOLINK (to be discontinued) is configured using special SIMOLINK blocks.

Starting with D7-SYS V6, communication links such as PROFIBUS DP, SIMATIC CPU ↔ FM 458-1 DP, as well as for SIMATIC TDC, can be also by configured with the help of new communication blocks.

The new blocks that are connected via special pointer interface can be used to access interface data from the CFC user interface.

Both of these configuration methods (virtual connections and pointer-based communication) can be used in parallel on the same hardware platform, in the same configuration, and even at the same interface.

3.11.4.1 Function principles

Message frame blocks (CRV_P, CTV_P as well as S7RD_P, S7WR_P) allow access to the received data blocks or the blocks to be sent (message frames) by providing a pointer to the respective data block.

This pointer is wired to read/write blocks (DRD..., DWR...). In combination with an offset definition, a write block is able save its input connection parameter to a selected position in the buffer. A read block accordingly fetches a parameter from the specified area of the receive buffer and makes it available at its output.

This basically means that a virtual interconnection is replaced with a (read/write) block and a "standard" CFC connection.

3.11.4.2 Applications

Large data volume

Pointer-based communication offers a particular advantage for handling very large amounts of data. This functionality is obviously more efficient in terms of the configuration, editing, and flexible interconnection of large amounts of data.

The CRV_P and CTV_P function blocks can be used with PROFINET, to transmit data in a data block either for data of a device or the data of the PIP (process image partition) for isochronous modules.

Access to the I/O area (P bus) at FM 458-1 DP

The blocks support transmission of 128 bytes from FM 458-1 DP to the S7-CPU and vice versa via the I/O area of the P bus.

The new S7RD_P/S7WR_P blocks can be used to copy all 128 bytes to a buffer using a block with optimized computing time. Flexible and indexed access to the buffer via the pointer interface is provided by means of read/write blocks.

Offset and length definitions also enable access to partitions.

Saving data to a data block

Data can be saved to a general-purpose data memory that can be accessed via pointer interface by means of read/write blocks. Several similar buffers can be created in this data block as a convenient means of saving and fetching data such as recipes.

3.11.4.3 Features of pointer-based communication

- Reduction of configuration efforts when creating CFCs, particularly if many virtual connections have to be created.
- New connections to frame data can be inserted and edited in online mode (pointers, buffer offset).
- Communication connections can be copied with or within chart blocks and be edited with these. This provides a particularly efficient means, for example, of configuring similar communication connections to a large number of drives.
- Indexed access to frame buffer data with the help of two offset definitions for the simple creation and usage of modular programs (e.g. chart blocks).
- Transparent processing (copying) of larger data volumes (e.g. block-oriented), e.g. using copy block CPY_P to copy data to data block DB_P.
- For FM 458-1 DP:
 - Transmission of large data volumes from the S7-CPU to FM 458-1 DP via C bus using "B-Receive" (BRCV).
 - Simple and efficient transmission of 128 bytes with low computing overhead via the I/O area of the P bus.
- A special read/write block is available for all data types (BYTE, INT, DINT, REAL).
- The type is verified prior to the access to REAL data.

For all platforms and interfaces of SIMATIC control systems

- These configuration options are basically available for all of the SIMATIC control system platforms, i.e. FM 458-1 DP and SIMATIC TDC, as block processing is independent of the sublevel hardware.
- For the same reason, it is always possible to use this type of block communication for all types of serial and parallel data transmission routes at which "virtual communication" is used nowadays.

3.11.4.4 Corresponding function blocks

The blocks that can be used are arranged under the family names "ZeigrKom" or "PointCom" in the CFC block catalog.

To facilitate identification and easy assignment to this block group, the blocks that feature functionality equivalent to existing blocks and now output a pointer for this application are appended a "_P" suffix (pointer) to their name.

Type name	Function
CPY_P	Copying buffer areas
CRV_P	Receive frame block (interface processing)
CTV_P	Transmission frame block (interface processing)
DB_P	Data block
DRD	Data Read REAL
DRD_D	Data Read DINT
DRD_I	Data Read INT
DRD_8	Data Read 8*REAL
DRD_8D	Data Read 8*DINT
DRD_8I	Data Read 8*INT
DRD_BY	Data Read BYTE
DWR	Data Write REAL
DWR_D	Data Write DINT
DWR_I	Data Write INT
DWR_8	Data Write 8*REAL
DWR_8D	Data Write 8*DINT
DWR_8I	Data Write 8*INT
DWR_BY	Data Write BYTE
S7RD_P	Receive 128 bytes via P bus (only for FM 458-1 DP)
S7WR_P	Transmit 128 bytes via P bus (only for FM 458-1 DP)
BRCV	Block data reception via S7 Connection (only for FM 458-1 DP)

3.11.4.5 Pointer interface

For pointer-based communication, a **pointer to the frame data buffer** is exchanged between the participating blocks:

This pointer is actually a pointer to a structure that contains information for monitoring purposes in addition to the pointer to user data, e.g. the sampling time, block class, byte/word swap. The pointer is assigned the "ZeigPuffer" connection comment.

3.11.4.6 Notes on configuration

- The frame blocks and read/write blocks must be assigned the **the same sampling time** in order to ensure consistency (verified during initialization).
- The **offset definitions** must be created with meticulous care.
 - a) For pointer-based communication, the configuration engineer must particularly observe the offset (in bytes) of the 16-bit value (INT) or 32-bit value (REAL, DINT) to be addressed.

The offset must always be less than the buffer size.

Prior to the access to buffer data, it is checked as to whether a range has been exceeded due to an incorrect setting of the offset length.

- If data is transmitted to a PROFIBUS DP station or SIMATIC CPU, you may need to swap the bytes (with INT) and words of the value to be transferred (with REAL, DINT).

For this purpose, the read/write blocks provide a "Swap" connection SWP.
- In order to transmit frames via interface, it is initially sufficient to define corresponding lengths only for the frame blocks (CRV_T, CTV_P and S7RD_P, S7WR_P). It is not yet necessary to configure read/write blocks. This procedure allows you to configure a test of the interface or of computing time load generated by the interface configured with minimum effort.

3.11.4.7 Examples based on CFC screenshots

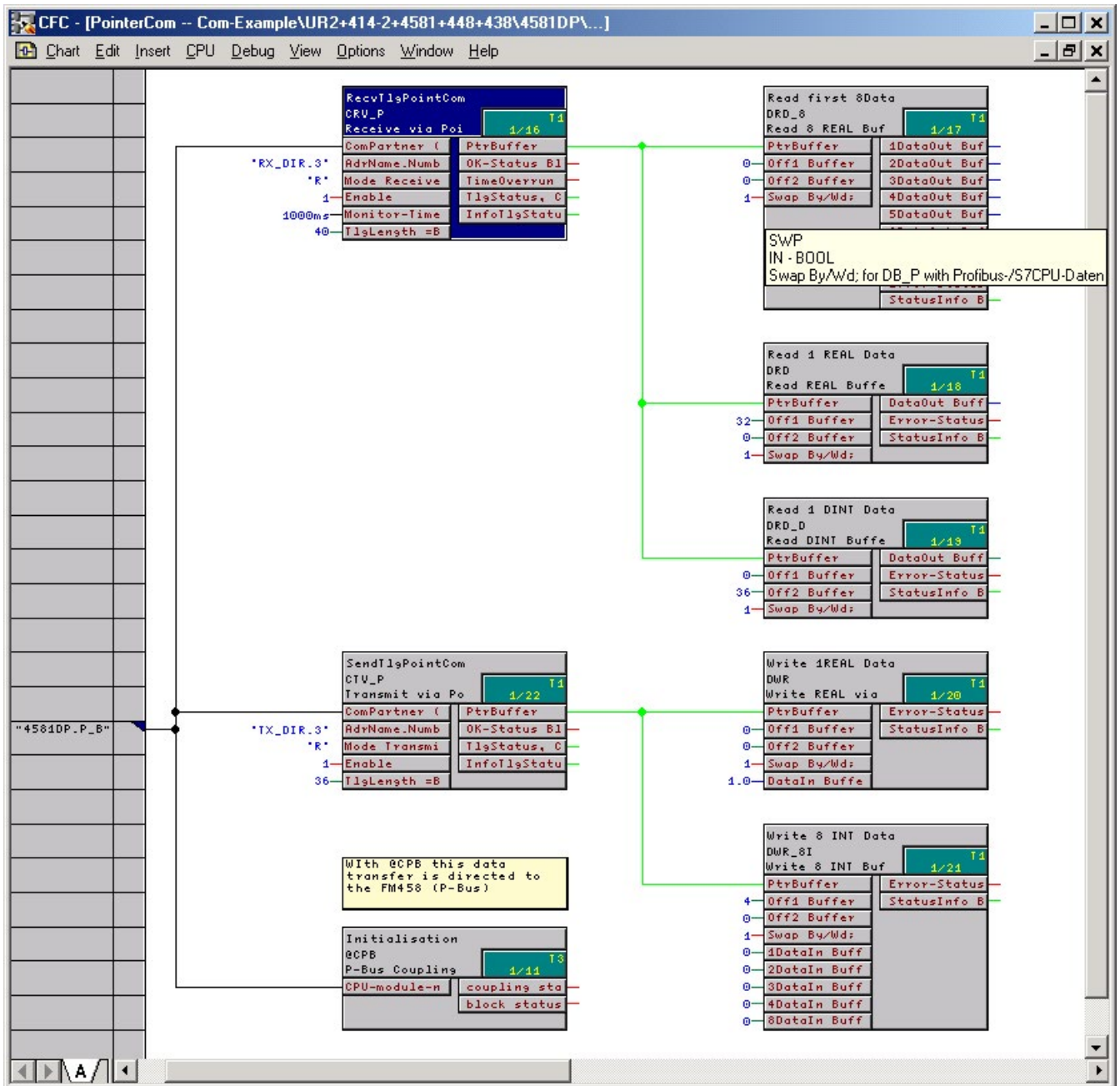


Figure 3-60 CFC screenshot: Data transmission with frame blocks and read/write blocks; here, for the P bus interface of FM 458-1 DP (@CPB); bytes/words must be swapped for data management on the SIMATIC CPU: SWP(Swap)=1

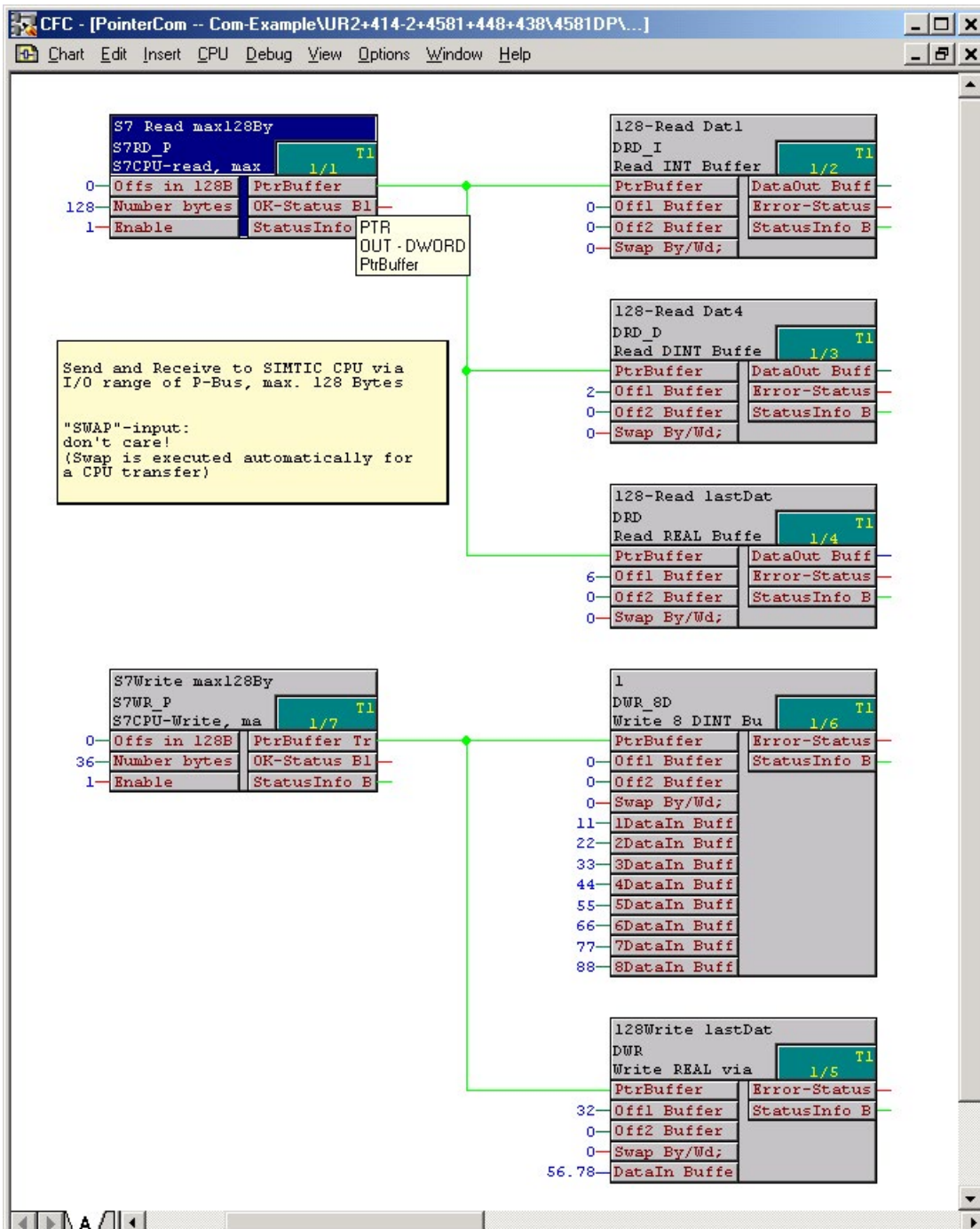


Figure 3-61 CFC screenshot: Data transmission SIMATIC-CPU ↔ FM 458-1 DP via P bus I/O area

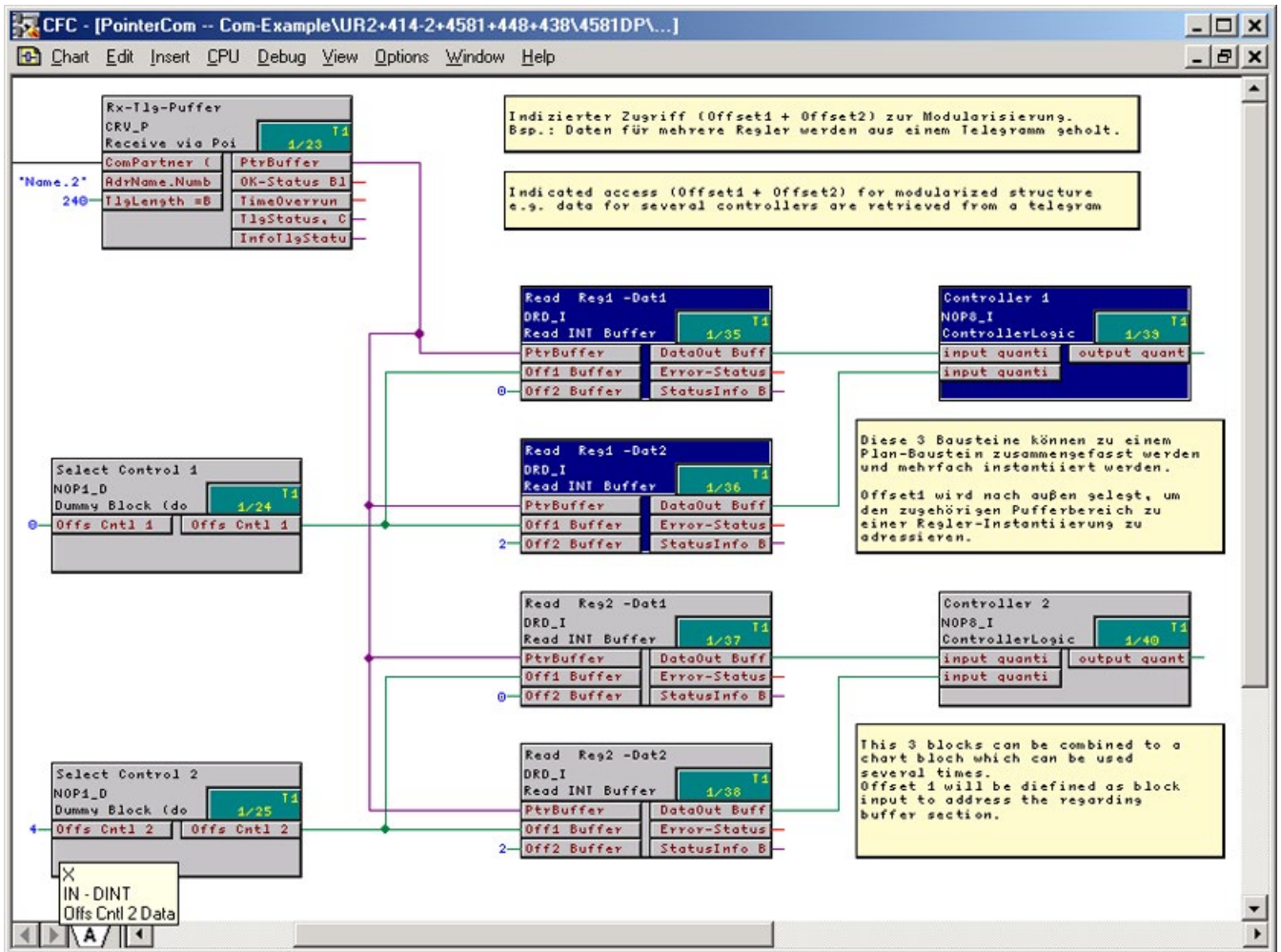


Figure 3-62 CFC screenshot: Indexed addressing of the frame data with two offset values

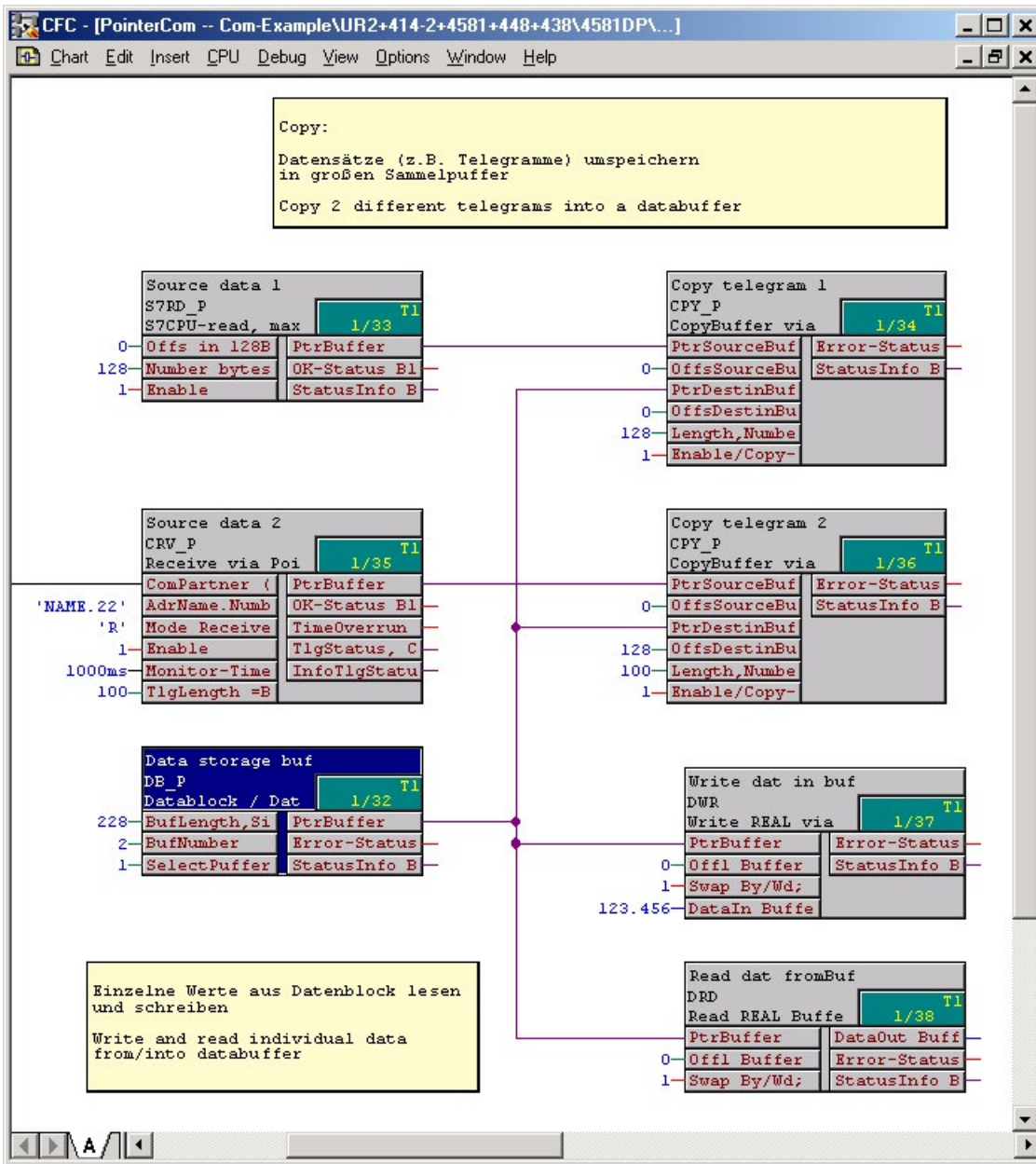


Figure 3-63 CFC screenshot: Saving two received frames to a different area in a data block and single access to data memory

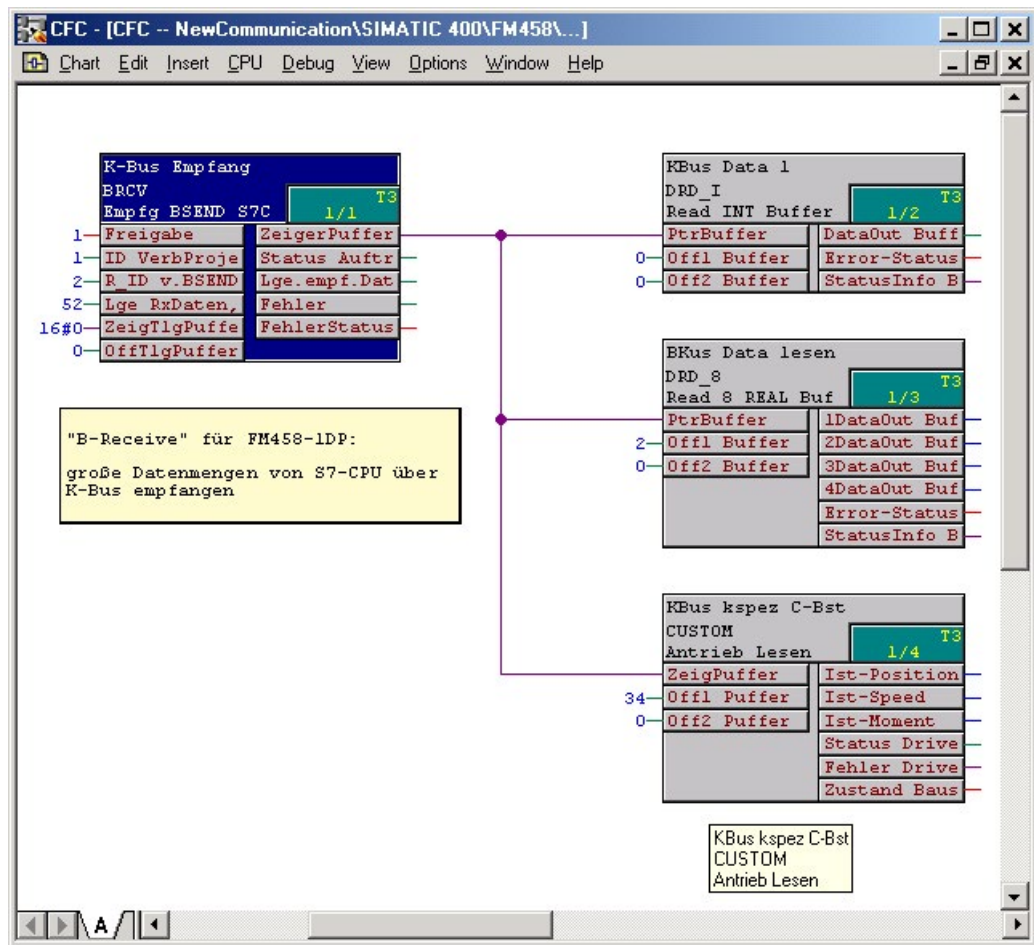


Figure 3-64 CFC screenshot: Large data volumes received from a SIMATIC CPU via K bus by means of BRCV

3.12 Communication utility service

Short description

- Provides a pool of information functions for user access to local system information on the CPU.
- Commissioning and debugging tool.
- Online download of user programs

Commissioning

In this phase, you display and/or edit configured data (setpoints/actual values) and optimize the configuration (modification of interconnections, or controller times).

Debugging

The tool supports you in locating sources of system malfunction (crash, startup problems) and disturbance which originate from the CPU module.

All activities of the communication service service are controlled by means of tasks received via coupling (in accordance with CTS and US I/O definitions).

HMI devices for the communication service service:

- Windows PC with CFC (e.g. in test mode)
- Windows PC with SIMATIC Manager

Local service

Via the local interface of a CPU (CPU555: PROFINET interface, CPU551: RS232 interface) can be accessed on a CPU. No additional configuration is required.

Note

Use the CFC and SIMATIC Manager to read the CPU module status.

You can find **more information** on the module status of the CPU in the system manual "D7-SYS - STEP 7, configuring CFCs and SFCs (<http://support.automation.siemens.com/WW/view/de/8776786/0/en>)" section "Diagnostics".

Central service

Each CPU of this rack can be accessed via MPI or Industrial Ethernet coupling you configured for the rack.

The following must be configured:

- One per rack:
 - MPI interface: One CP50M1 module and a central block for MPI coupling @MPI
 - Industrial Ethernet coupling: CP51M1 module and a central block @TCPIP
- At least once per CPU551:
 - SER service function block.

You can find **more information** on the MPI coupling in the section "MPI coupling (Page 145)" and more information about Industrial Ethernet coupling in the section "TCP/IP coupling (CPU555; CP51M1) (Page 123)".

Note

CPU555

A CPU555 also enables central service without the SER function block. In other words, you can access any CPU555 from any other CPU555 without having to configure an SER.

3.12.1 SER function block

Definitions at the connections

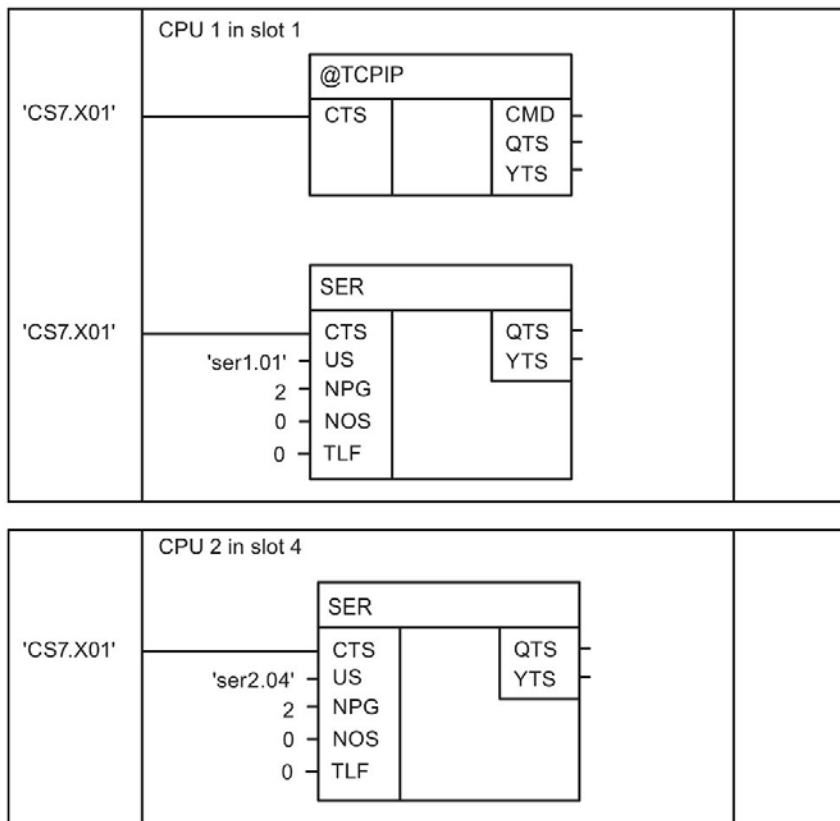
The "SER" function block has an I/O for coupling, and can be configured several times for each CPU.

The **CTS** I/O defines the coupling module and interface used to connect an HMI device.

The channel name and address level 1 are specified at the **US** I/O

- **Channel name**
 - Max. 6 characters
 - ASCII characters except "period" and @Channel name must be unique on a data interface
- A dot "." is appended to the channel name
- **Address level 1**
 - CPU slot number. The operator program uses this number to address the CPU.
 - This number must be unambiguous: e.g. "01".

Example: Configuration with CFC



3.12.2 System load, response times

General

The service is actually processed within a sampling time of approximately 32 ms (the sampling times specified at the SER blocks are therefore not decisive for processing). Within the sampling time used, the service blocks are allocated a certain computing time slice that corresponds to a maximum of one basic clock cycle (T0).

Note

The ratio of the basic clock cycle T0 to the sampling time used determines CPU performance and system load.

Example1

Basic clock cycle T0=1ms; selected sampling time=32ms

- A time slice of 1 ms is reserved for the service utility in each 32 ms interval
- System load=1 ms / 32 ms=0.03125=3.125 %

Example 2

Basic clock cycle T0=2ms; selected sampling time=16ms

- A time slice of 2 ms is reserved for the service utility in each 16 ms interval
- System load=2 ms/16 ms=0.125=12.5 %

Computing time

The computing time available is used by all service blocks at the same priority level. This means that all SER blocks are executed once if possible within the time available. An SER block processes only one task per clock cycle. Remainders of reserved computing time slices not used to the full extent are made available to the system, e.g. if no task execution is pending.

Distribution of resources

For multiple configuration and simultaneous access to system resources which are only available once (e.g. EEPROM), the first to request a resource is the first to receive it. All others are rejected and an error message is output, at the latest after 1 s ("resource occupied").

3.13 Communication utility time synchronization

General

The communication utility time synchronization provides a uniform system to several SIMATIC TDC racks.

Time

The time can be obtained from:

- An NTP time signal
- The leftmost CPU plugged into a rack
- A time server via the Industrial Ethernet coupling module, CP51M1

The time is distributed as follows:

- Within a rack via a coupling memory module
- To other racks via the rack coupling (CP52A0 / CP53M0)

Function block

Exactly one function block RTCM must be configured on each rack for distribution of the system time.

You can find **more information** on the configuration of function blocks in the programming manual "SIMATIC D7-SYS Selecting function blocks (<http://support.automation.siemens.com/WW/view/de/14952400/0/en>)".

The following function blocks serve to read the system time:

- RTCABS: Absolute time in date/time format
- RTCREL: Relative time in seconds since 01.01.88

These blocks can be configured as required.

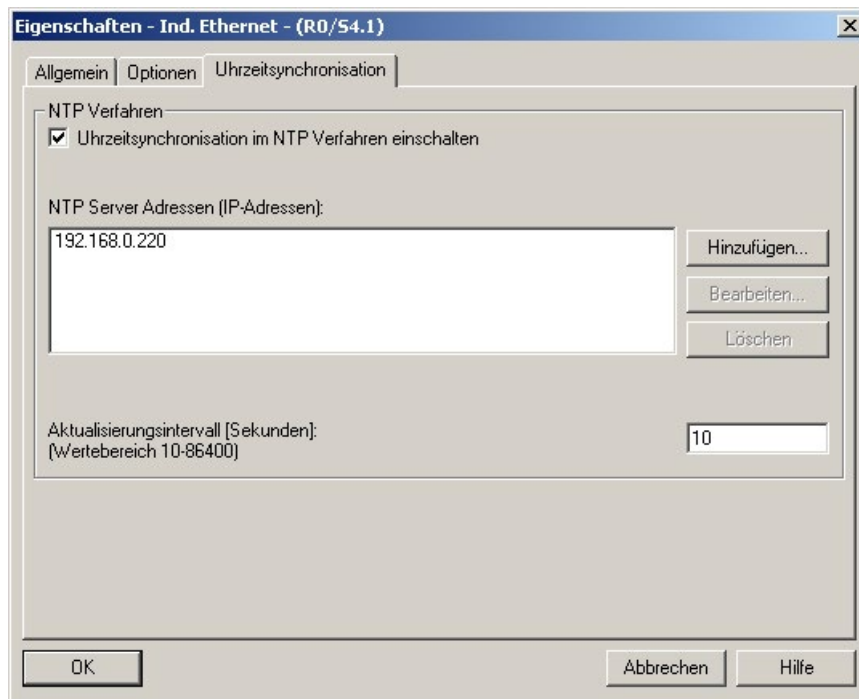
CP51M1

The CP51M1 IE module is capable of actively fetching the time from up to four NTP time servers (e.g. SICLOCK TM).

For more information on the NTP time synchronization process, refer to: www.ntp.org

You need to make the following settings in HWConfig, "Time synchronization" tab of "Ind. Ethernet connection of CP51M1":

- Activate the NTP process
- Add the IP address of the NTP server(s)
- Set the update interval (seconds)



The CP51M1 module fetches the current time from all configured NTP servers within the update interval (10 seconds is recommended). The time returned from all configured NTP servers may not differ noticeably ($\ll 1$ ms), as the time of the currently "best" NTP server is always used. Time differences between the NTP may otherwise lead to time jumps. The "best" NTP server is also continuously determined based on the time request runtime.

Note

If more than one NTP server is configured, only the NTP servers that are calibrated automatically based on a reference clock signal (e.g. DCF77 or GPS) within the last 24 hours will be used for time synchronization. If only one NTP server is configured, this server will be used even if it was not synchronized with a reference clock.

Important events are written to the CP51M1 diagnostics buffer, e.g.:

- Synchronization with NTP server (incoming / outgoing)
- NTP server is not accepted
- Loss of synchronization of the NTP server with reference clock

Note

If an SNTPR-FB has been configured for this CP51M1 and an additional NTP, the time determined by SNTPR is broadcast to the TDC rack. The time determined using the NTP method is only used to set the clock of the CP51M1 module. A corresponding entry is made in the diagnostics buffer.

3.14 WinCC connection to SIMATIC TDC via standard channel (SIMATIC S7 Protocol Suite.CHN)

The chapter describes the procedures in WinCC for configuring access to process variables (block connections) of a SIMATIC TDC CPU via "standard channel". The following figure illustrates the different configuration and coupling options.

This chapter provides all information with regard to coupling over TCP/IP by means of the FB property "OCM" functions (Operator Control and Monitoring) during the creation of an address book.

The following chapter describes special features of the configuration variant with function block "S7DB_P" that can be discarded when creating an address book.

This is followed by information related to the use of MPI and PROFIBUS DP coupling types. (However, when using PROFIBUS DP coupling for visualization, you should note that this coupling type, which is generally used for high-speed applications, is possibly slowed down by the HMI signals.)

In the final paragraphs you are provided information on the use of the D7-SYS-OS engineering tool (WinCC mapper).

WinCC – SIMATIC TDC

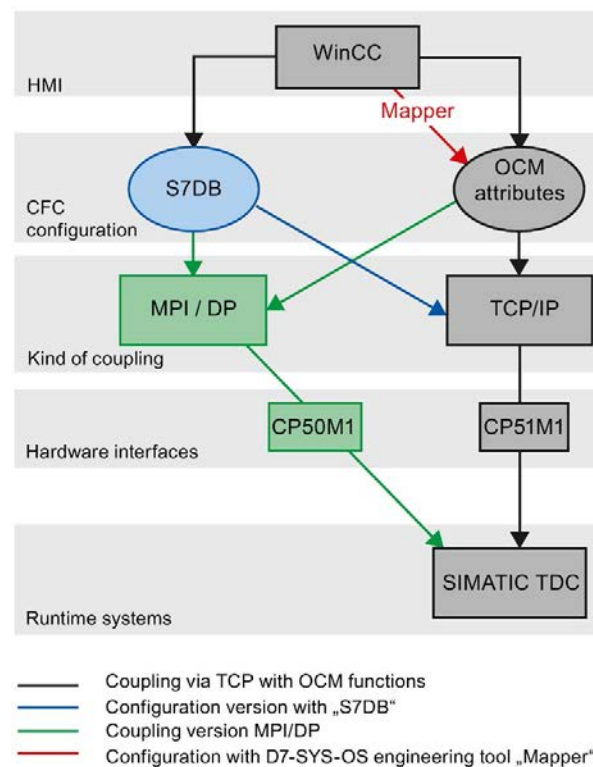


Figure 3-65 Overview of communication and configuration options

Note

Increased communication load

Depending on settings, PG/PC tools such as WinCC or CFC online and HMI Panels create an increased communication load.

This can result in longer response times and communication failures.

3.14.1 Coupling over TCP/IP with "OCM" functions

This section describes the procedure in WinCC for configuring access to process tags (block I/Os) of a SIMATIC TDC CPU.

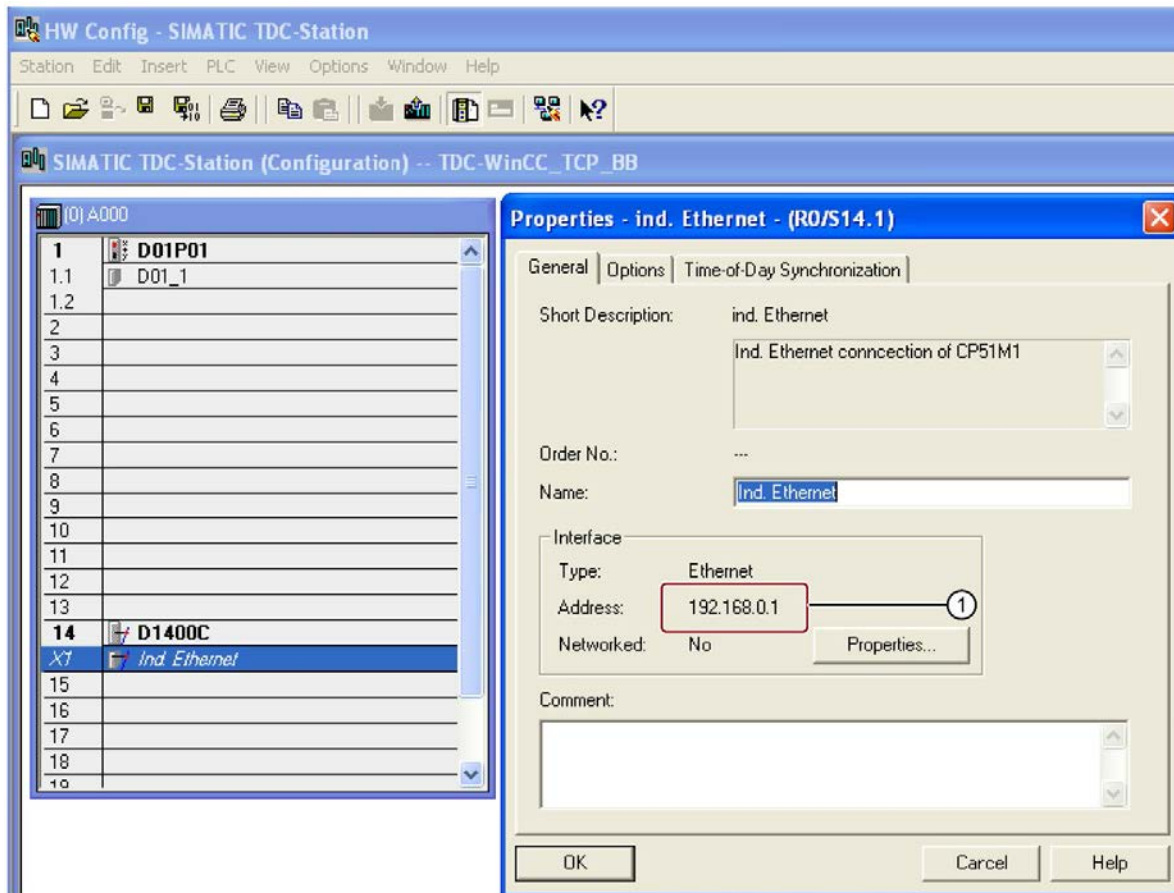
Note

CPU555 PROFINET interface or CP51M1 communication module

Both the PROFINET interface of the CPU555 and a CP51M1 communication module can be used for the TCP/IP connection, as described in the following example.

3.14.1.1 Configuring the coupling-relevant TDC hardware

A CP51M1 must be configured as communication module for TCP/IP coupling in the SIMATIC TDC rack. Click on the "Properties" button to open an additional window for inserting a subnet and setting the IP address.



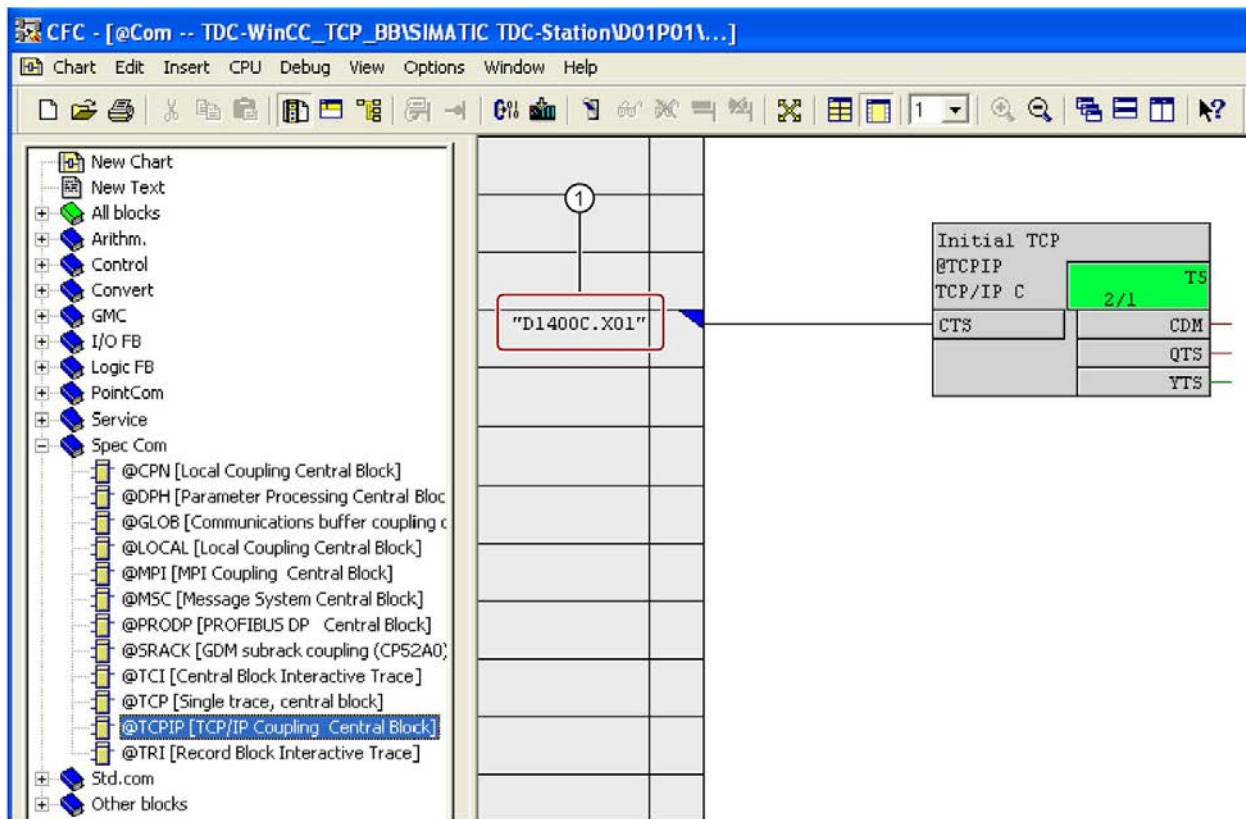
① IP address

Figure 3-66 Set IP address

3.14.1.2 CFC configuration

Configuring the coupling-relevant CFC function blocks

Configuration of a function block for initialization of the TCP/IP interface ("X01") of module CP51M1 in slot 14 ("D1400C"). The function block must be configured in a task between 32 and 256 ms.

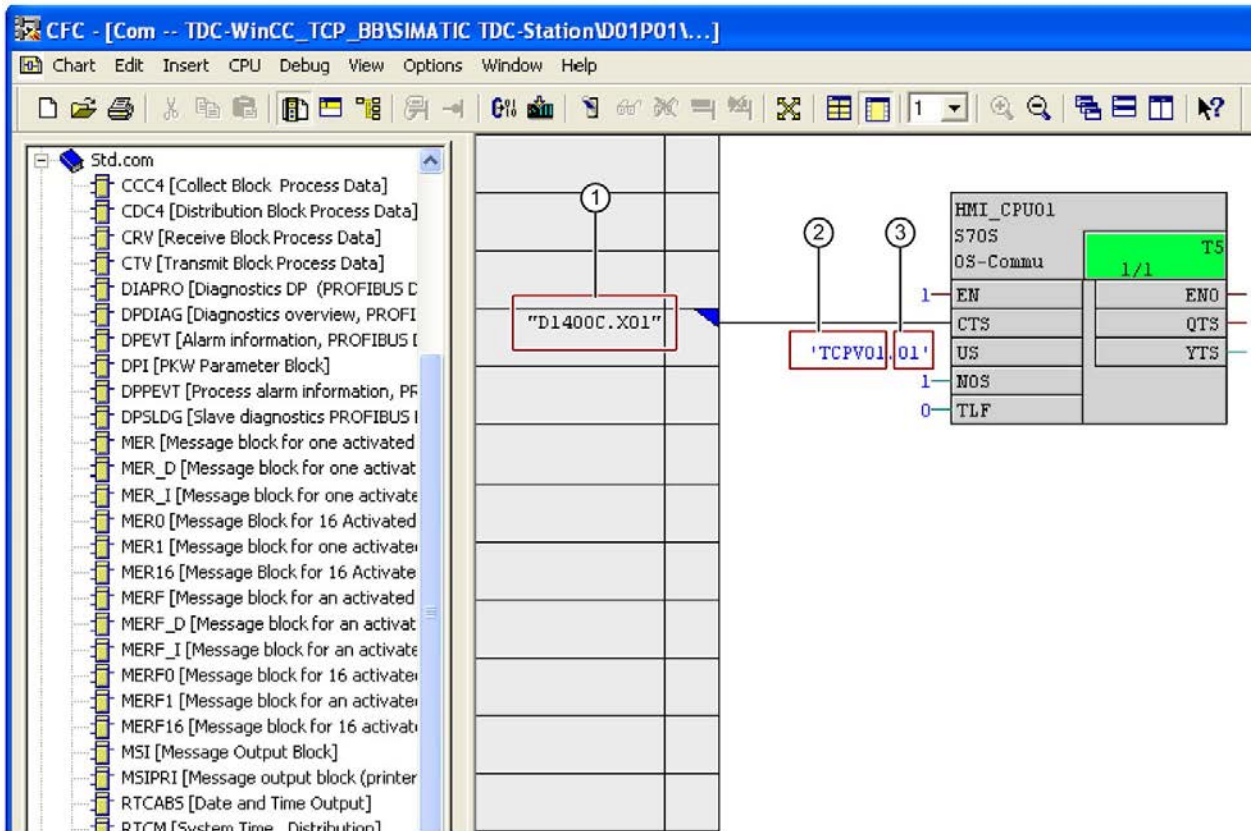


① "Front connector receptacle" on the communication module

Figure 3-67 Initialization of the TCP/IP interface

3.14 WinCC connection to SIMATIC TDC via standard channel (SIMATIC S7 Protocol Suite.CHN)

Configuration of the "S7OS" function block for initialization OS communication on "CPU 01" in slot 1. You could basically use the "SER" block instead of the "S7OS" block, but you would have to expect a negative impact on response times. For more information, you should refer to the function block description.



- ① "Slot.Front connector" of the communication module
- ② Unique name with a length of 6 digits
- ③ Slot number of the CPU

Figure 3-68 Initialization of the OS communication

Marking the function block connections in the CFCs and creating the address book

To identify block connections that must support OCM in WinCC, you first have to mark the corresponding objects in the CFCs. Take the following steps:

1. Open the Properties dialog of the block, set the check mark for "OCM supported" and then press the "OCM" button (see the following figure).

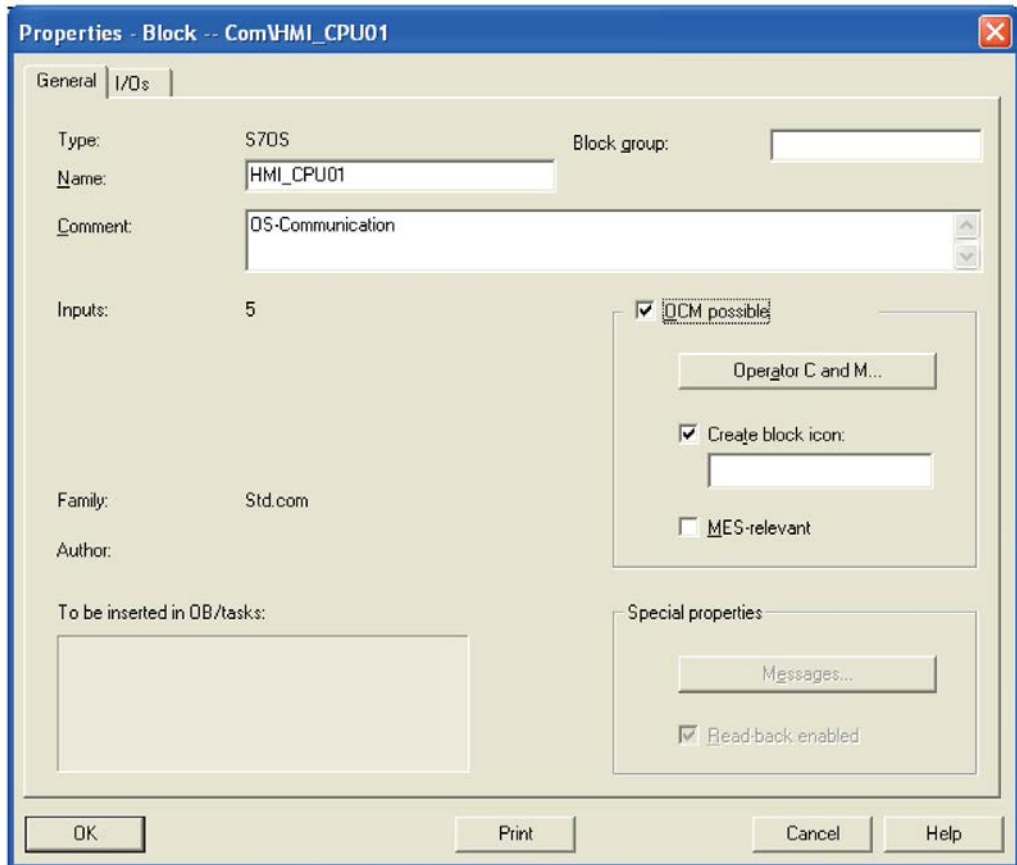


Figure 3-69 Set OCM

2. In the next dialog, set the check mark for "Complete block structure" if all I/Os of the selected block must support OCM (see the following figure). If you only need to select individual connections, skip this step and continue with Step 3.

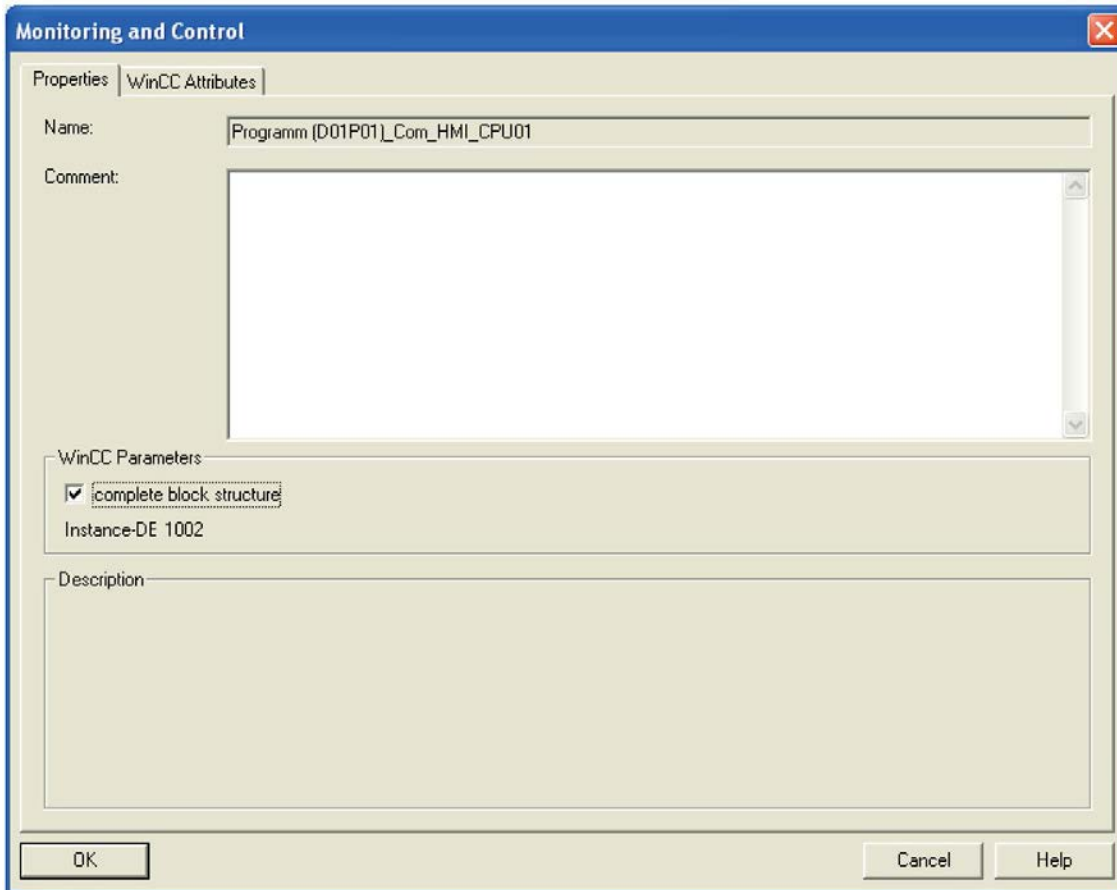


Figure 3-70 Log on all blocks for OCM

3. Mark the connections that support OCM individually in the WinCC attribute tab if you do not want to select all connections as in Step 2 (see the next figure).

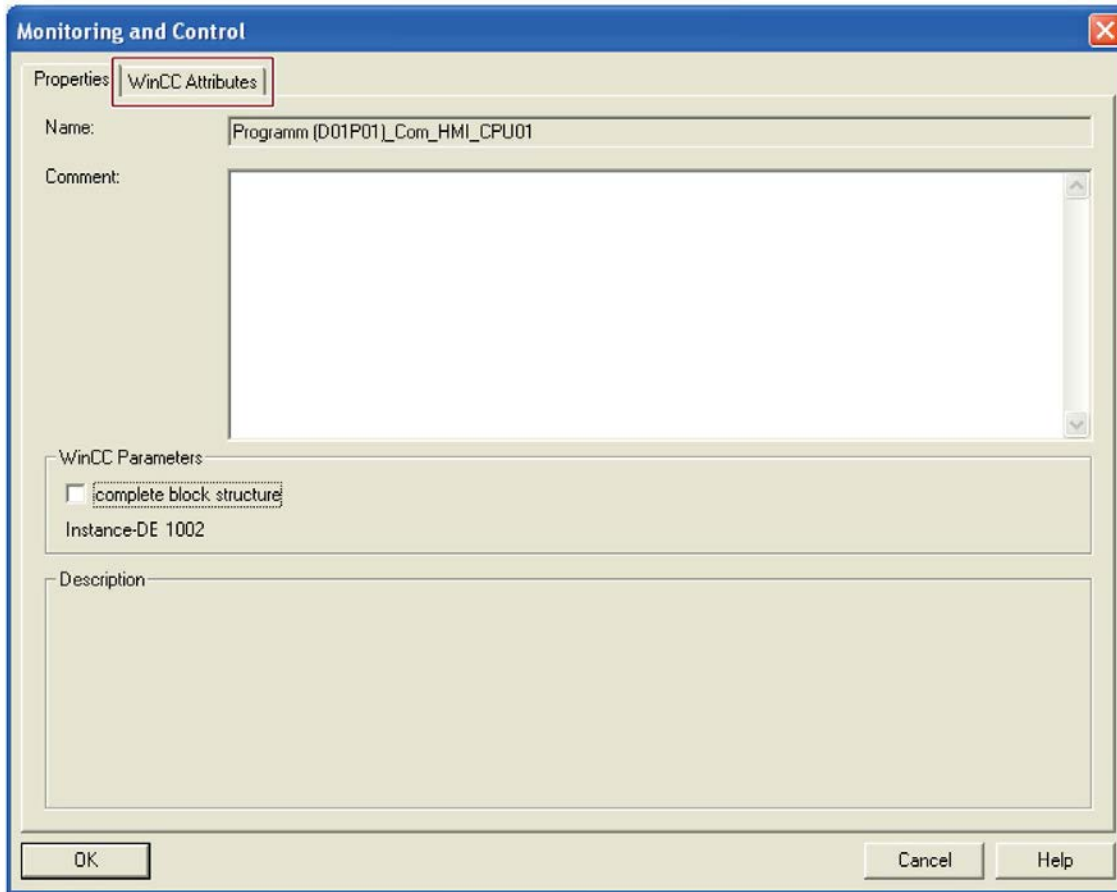


Figure 3-71 WinCC attributes -> Log on individual I/Os

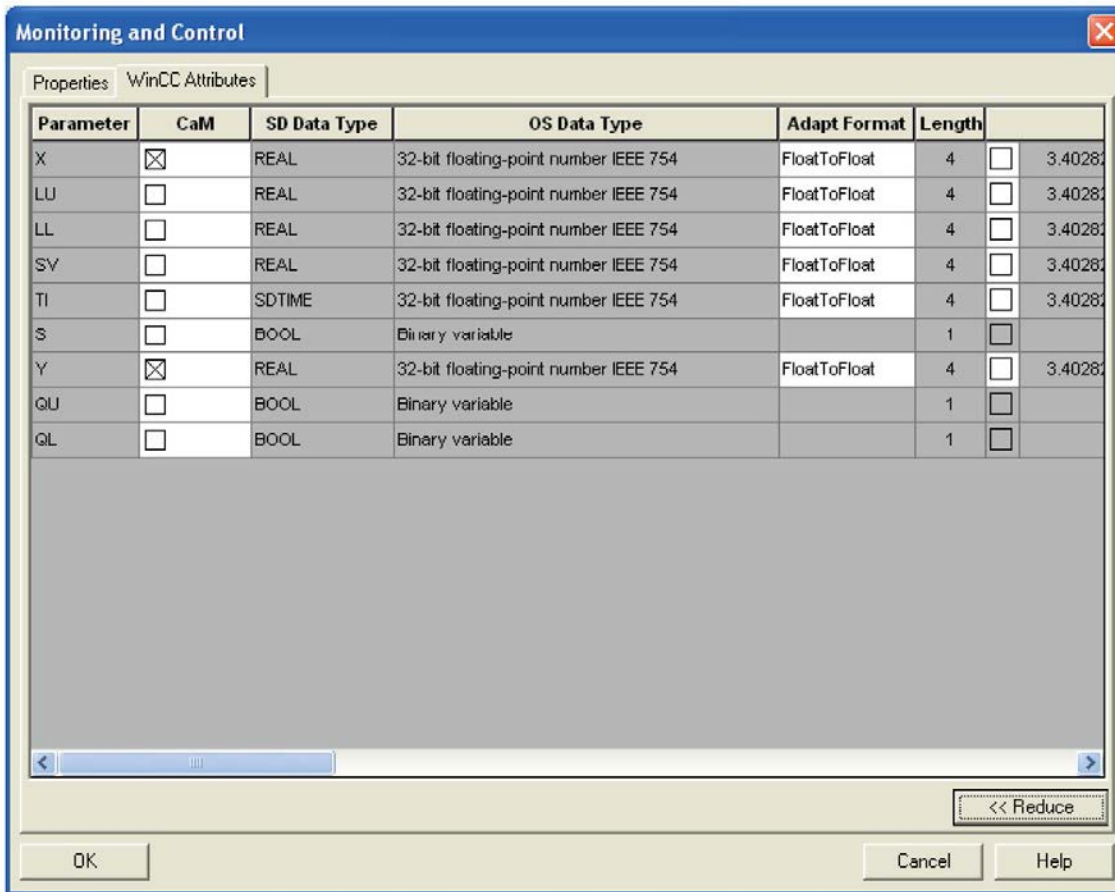


Figure 3-72 Log on individual I/Os

4. Repeat steps 2 and 3 for all blocks that are to be controlled and monitored.
5. Select the address book creation in the options dialog for compilation from CFC (Options > Customize > Compile/download) (see the next figure) in order to obtain the address information for the WinCC configuration.

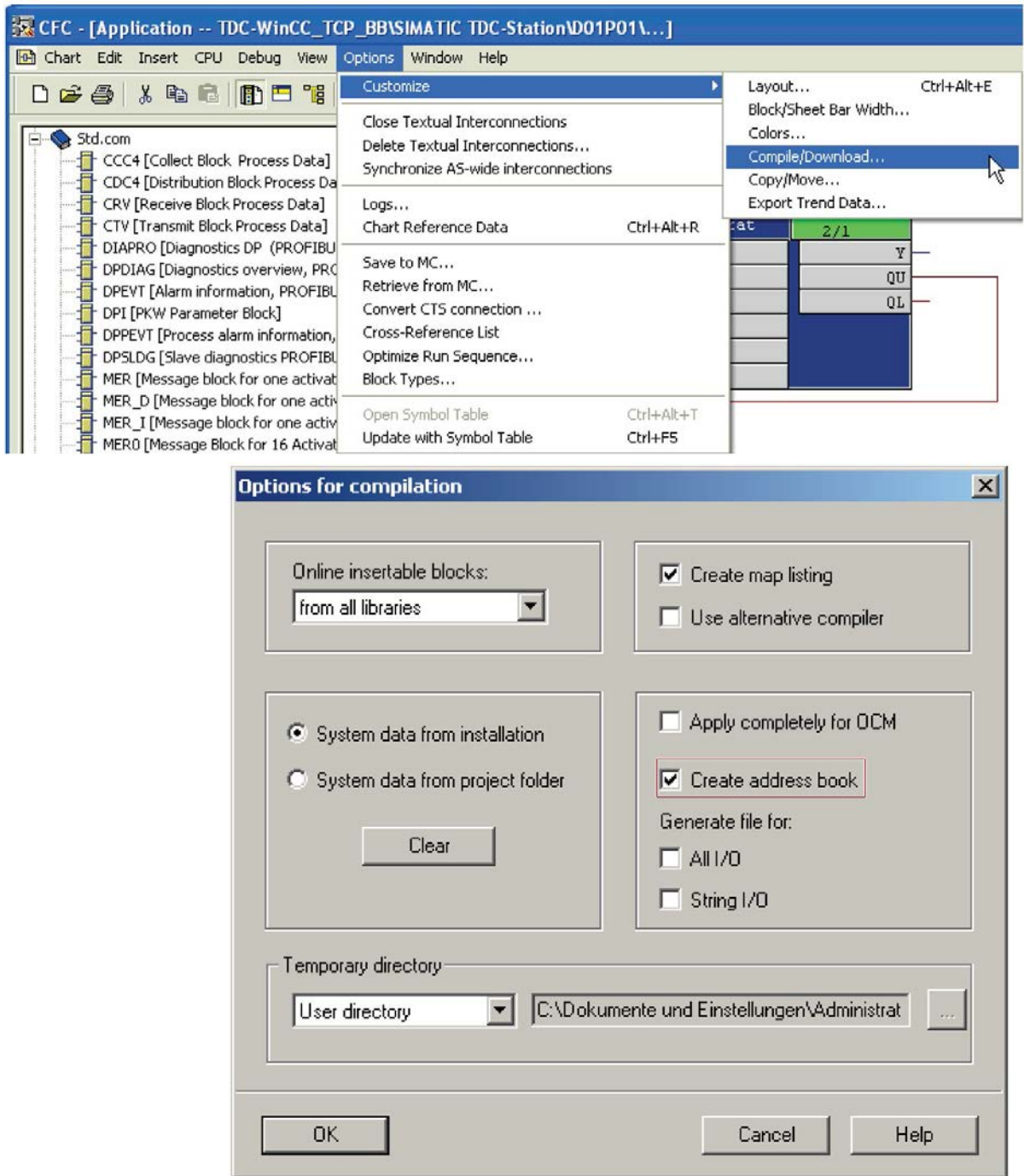


Figure 3-73 Create address book

The address book is created during compilation, which means that all necessary activities in the CFCs have been completed. Access to the configuration in WinCC is always possible on completion of the download to the target system.

A "D7-SYS-OS Engineering Tool" that is referred to as "mapper" as of herewith is included in your D7-SYS V7.1 package. When executed, the tool sets a tag (WinCC variable) for each selected function block connectors.

If you are using the mapper, you can leave the workflow at this point and continue with section Configuration using the "D7-SYS-OS Engineering Tool" (Page 268).

The DB numbers and offsets of the connections that are necessary for the WinCC configuration can now be fetched from the address book. Select the **"Options >Logs"** menu command to search for a log that contains the storage location of the address book.

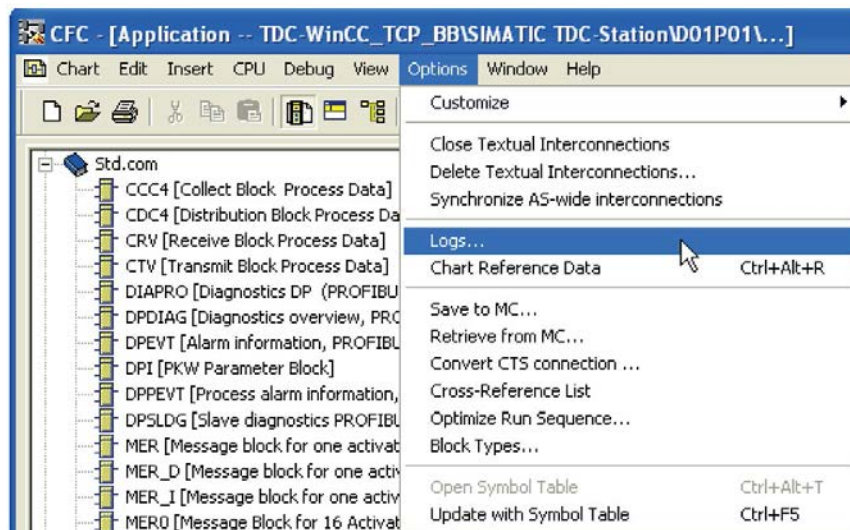


Figure 3-74 Select protocol

The storage location of the address book is marked in the following view of the log:

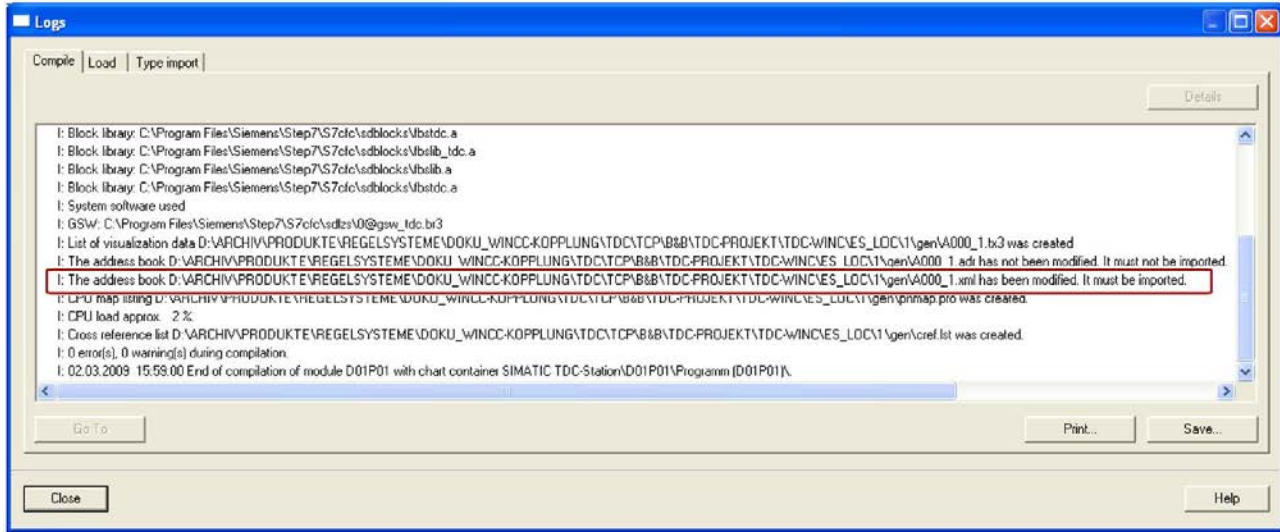
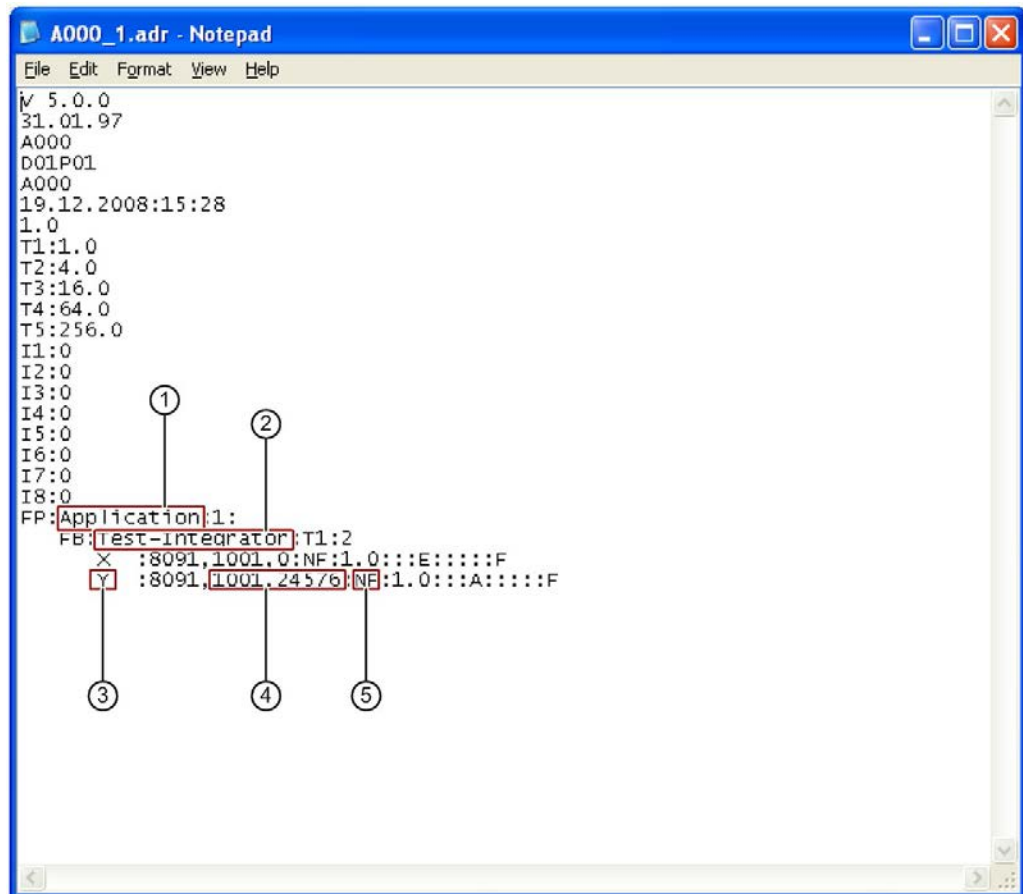


Figure 3-75 Storage location of the address book

WinCC-relevant information such as data block numbers and offsets of the selected function block connectors can now be fetched from the address book.



- ① Function chart name
- ② Function block name
- ③ Connector name
- ④ Data block number (DB), offsets
- ⑤ Data type

Figure 3-76 Refer to WinCC-relevant information

3.14.1.3 WinCC configuration

Proceed as following to create the WinCC configuration:

1. Start the WinCC Control Center.
2. Create a new project or open an existing one.
3. Select **Tag management > right-click > Add new driver > SIMATIC S7 Protocol Suite.CHN > Open** to create a new driver. Continue with the next step if the driver already exists.
4. Select **TCP/IP > right-click > New connection** to create a new connection.

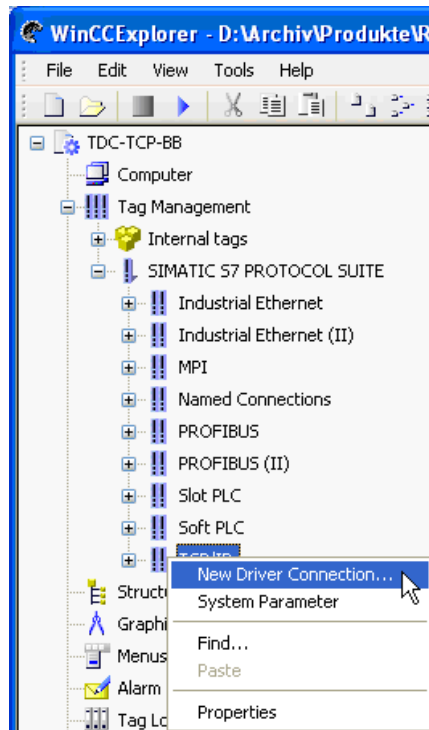


Figure 3-77 Create new connection

- Assign a name to the connection in the dialog, press the Properties button and enter the connection parameters (TCP/IP address and slot can be fetched from HWConfig; see the following figure).

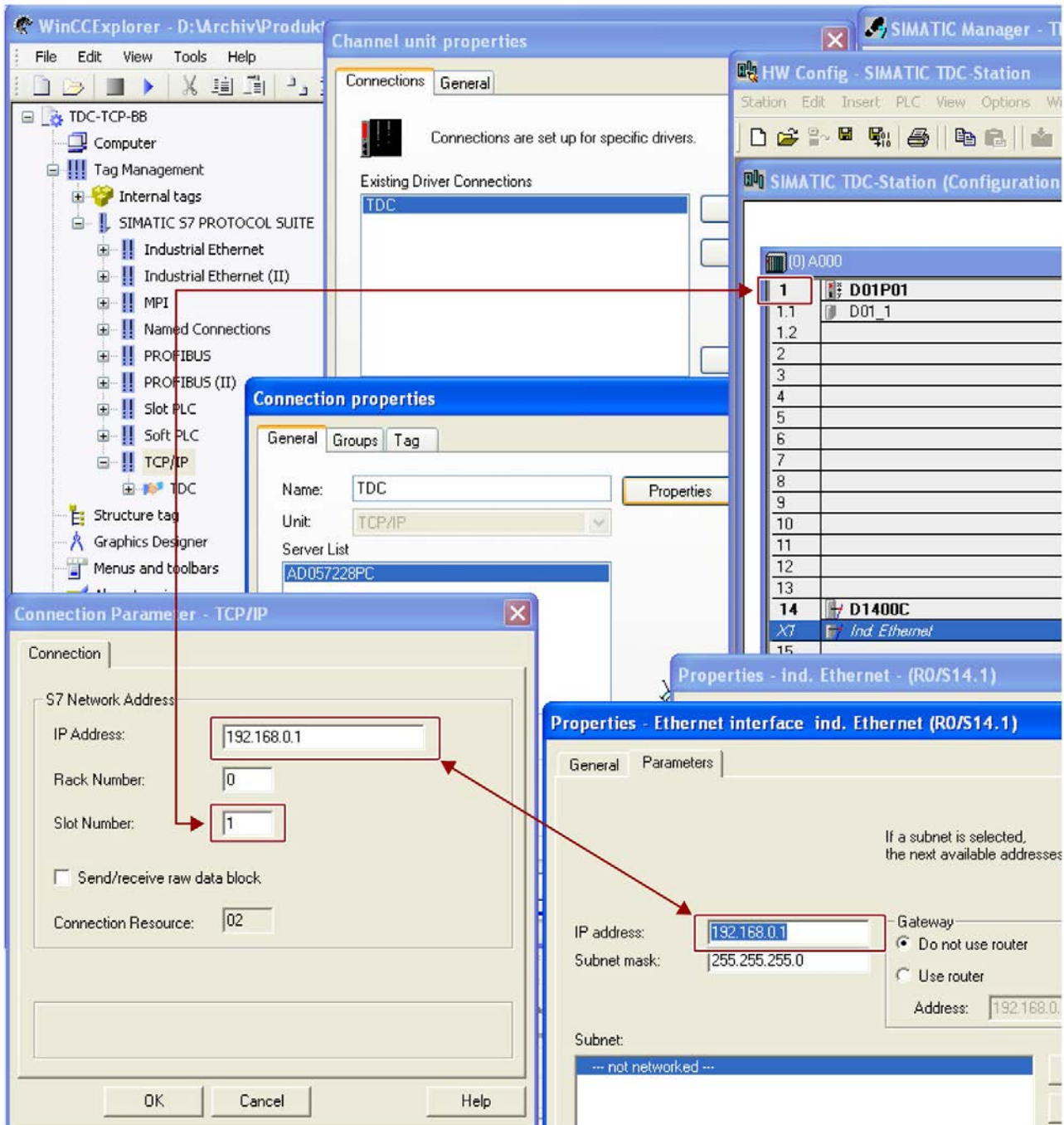


Figure 3-78 Assign name and set connection parameters

A new TCP/IP connection is created after you exit the dialog with OK.

6. **Creating the tags:** Right-click the connection you have created in the previous step and select "New tag" from the shortcut menu.

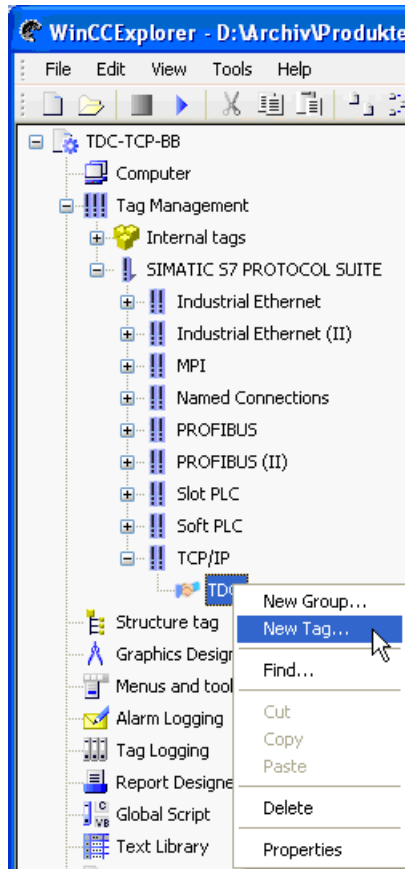


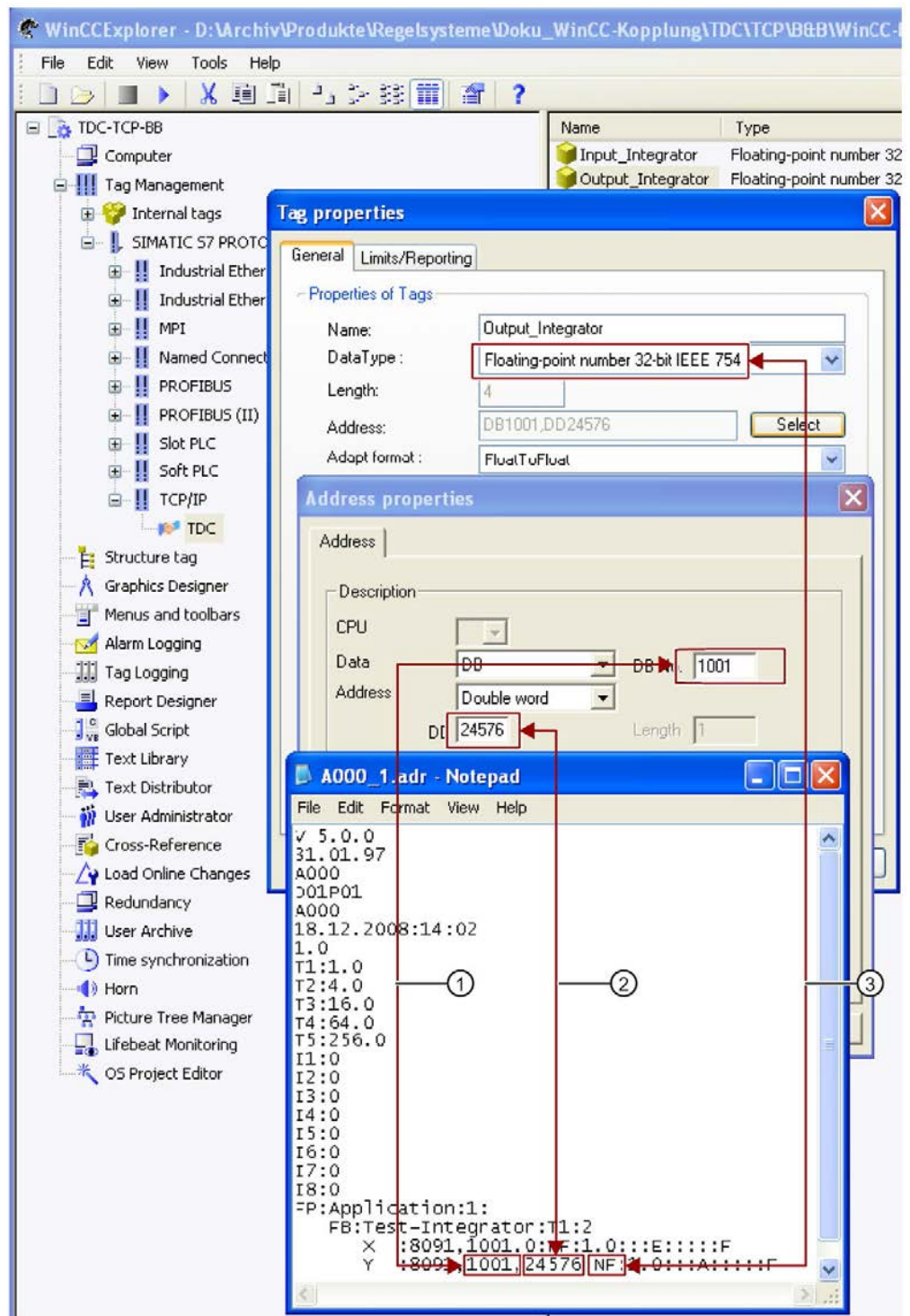
Figure 3-79 Create tag

7. Enter a tag name in the next dialog (e.g. function block name_I/O name, or any other name).

The data type fetched for the selected connector from the CFC configuration, or from the address book created during CFC compilation, can be configured under "Data type" (a reference table of the data types is available below).

Press the "Select" button to open the address dialog. Specify the DB number and offset in this address dialog. Fetch this data for the respective I/O from the address book that was generated during CFC compilation (see the next figure).

3.14 WinCC connection to SIMATIC TDC via standard channel (SIMATIC S7 Protocol Suite.CHN)



- ① DB number
- ② Offset
- ③ Data type

Figure 3-80 Enter data

STRUC V.4.x data type	D7-SYS data type	Designation
B1	BO	BOOL
I2 / N2 / O4	I	Integer
I4 / N4 / O4	DI	Double-Integer
NF	R	Real
V1	BY	Byte
V2	W	Word
V4	DW	Double-Word
NS	S	String
TF	TS	SDTime
IK, NK, CR, MR, TR, RR	GV	Global

8. Repeat the procedure as of step 6 if you need additional block connections.
9. The check mark for "Use cyclic read services in the AS" may **not** be set (see the next figure) in the "System parameters" input field for drivers (TCP / MPI > right-click > System parameters).

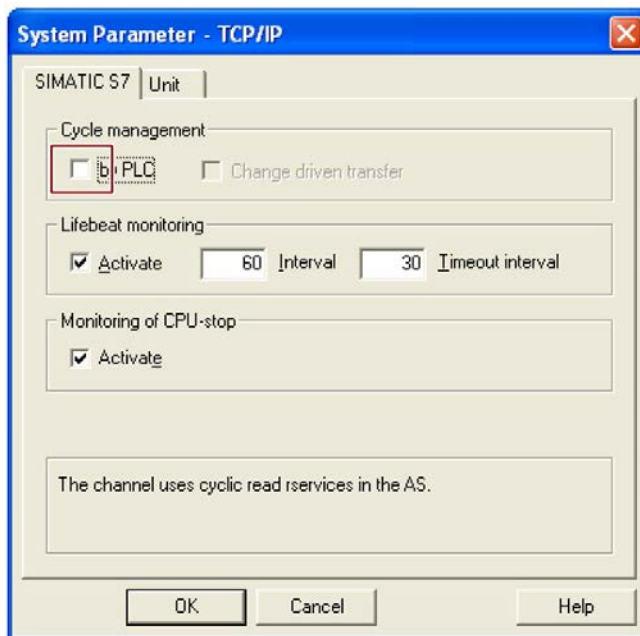


Figure 3-81 Disable the cyclic read services on the AS

10. After you have entered the corresponding data and closed the dialogs with OK, a tag is created in WinCC for the selected block I/O.

11. Select the "Logical device name" as well in the "System parameters" dialog.

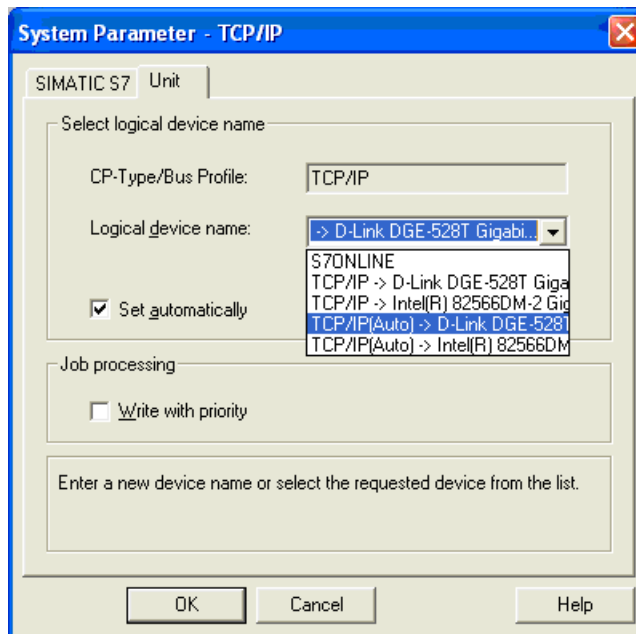


Figure 3-82 Select the logical device name

12. You can now set the references to your tags in the screen configurations.

3.14.2 "S7DB" configuration variant

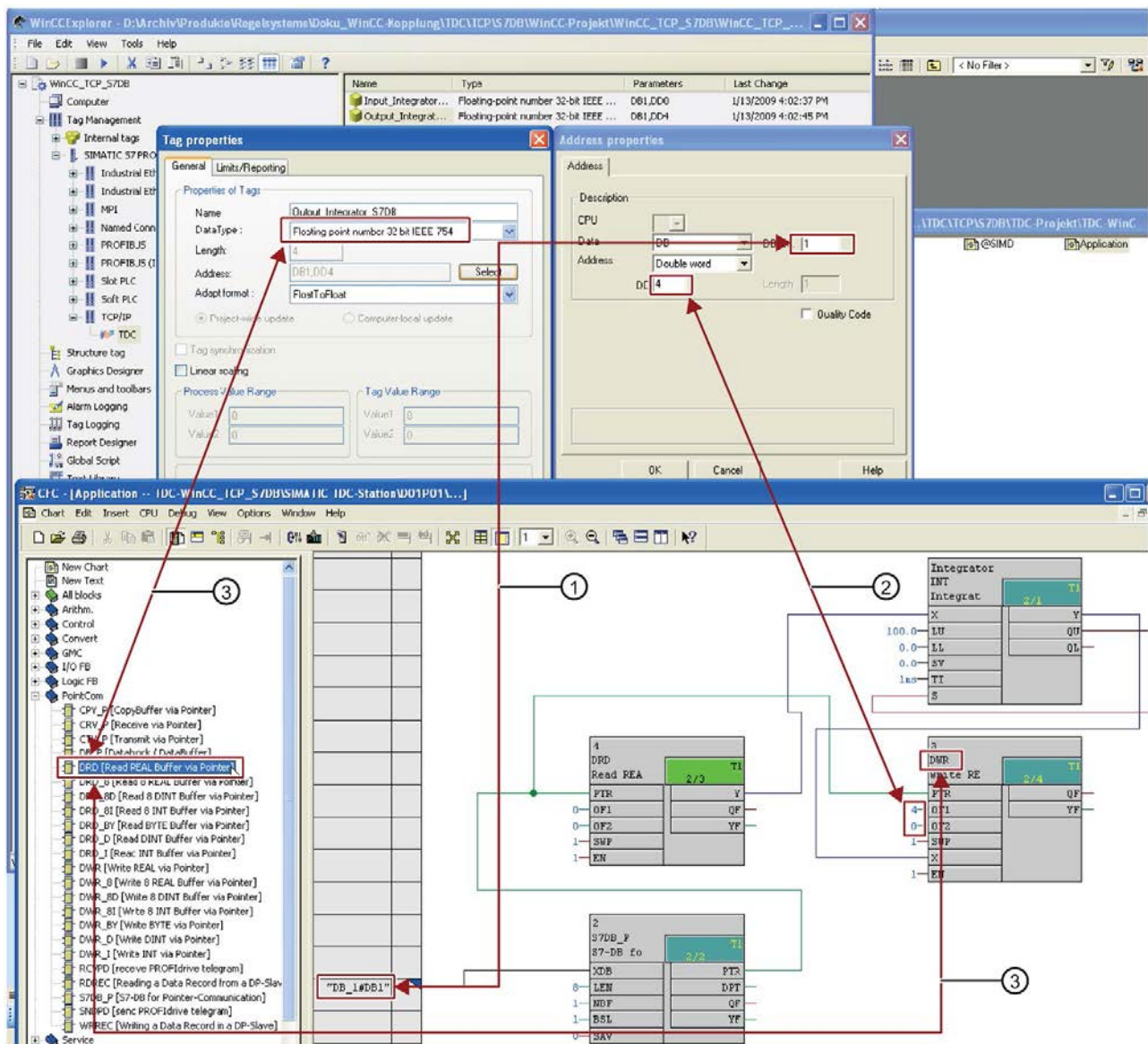
If you are using the "S7DB_P" function block, you can dispense of connection marking in the CFCs and creation of the address book.

Instead, configure corresponding pointer blocks for use with function block "S7DB_P". The block connections to be accessed in WinCC must be wired to function block S7DB_P by means of "pointer blocks". Function block S7DB_P creates a data block for this data.

Proceed as follows:

1. Configure function block S7DB_P. Right-click in connector "XDB", select "Interconnection to operands" and specify the DB number.
2. Wire all connectors to be visualized (e.g. as in the next screen: Connectors "X" and "Y" of function block "Integrator") to a user-defined pointer block that belongs to the function block family "Pointer Com", (e.g.: function block "DRD" for reading variables of type REAL) and corresponds to the respective connector type.
3. Wire connector "PTR" of S7DB_P to the "PTR" connectors of all pointer blocks.
4. Fetch the data block numbers (e.g.: DB1) from the sheet bar and the offset from the pointer block connector (OF1 + OF2) of the CFC configuration. Press the "Select" button to open the address dialog and enter the DB number and offset. (See figure below) Save your settings and close the dialog with OK.

3.14 WinCC connection to SIMATIC TDC via standard channel (SIMATIC S7 Protocol Suite.CHN)



- ① DB number
- ② Offset
- ③ Data type

Figure 3-83 Enter data

All additional configuration tasks do not differ from the procedures for using OCM functions.

3.14.3 MPI and PROFIBUS DP coupling variants

The following chapter describes deviations of MPI/DP coupling compared to the TCP coupling that you must take into account.

3.14.3.1 Hardware configuration

The following example shows a CP50M1 that is to be assigned the corresponding coupling type for PROFIBUS DP or MPI coupling.

1. Select and double-click the CP50M1 interface.
2. Select the "General" tab and click "Properties".

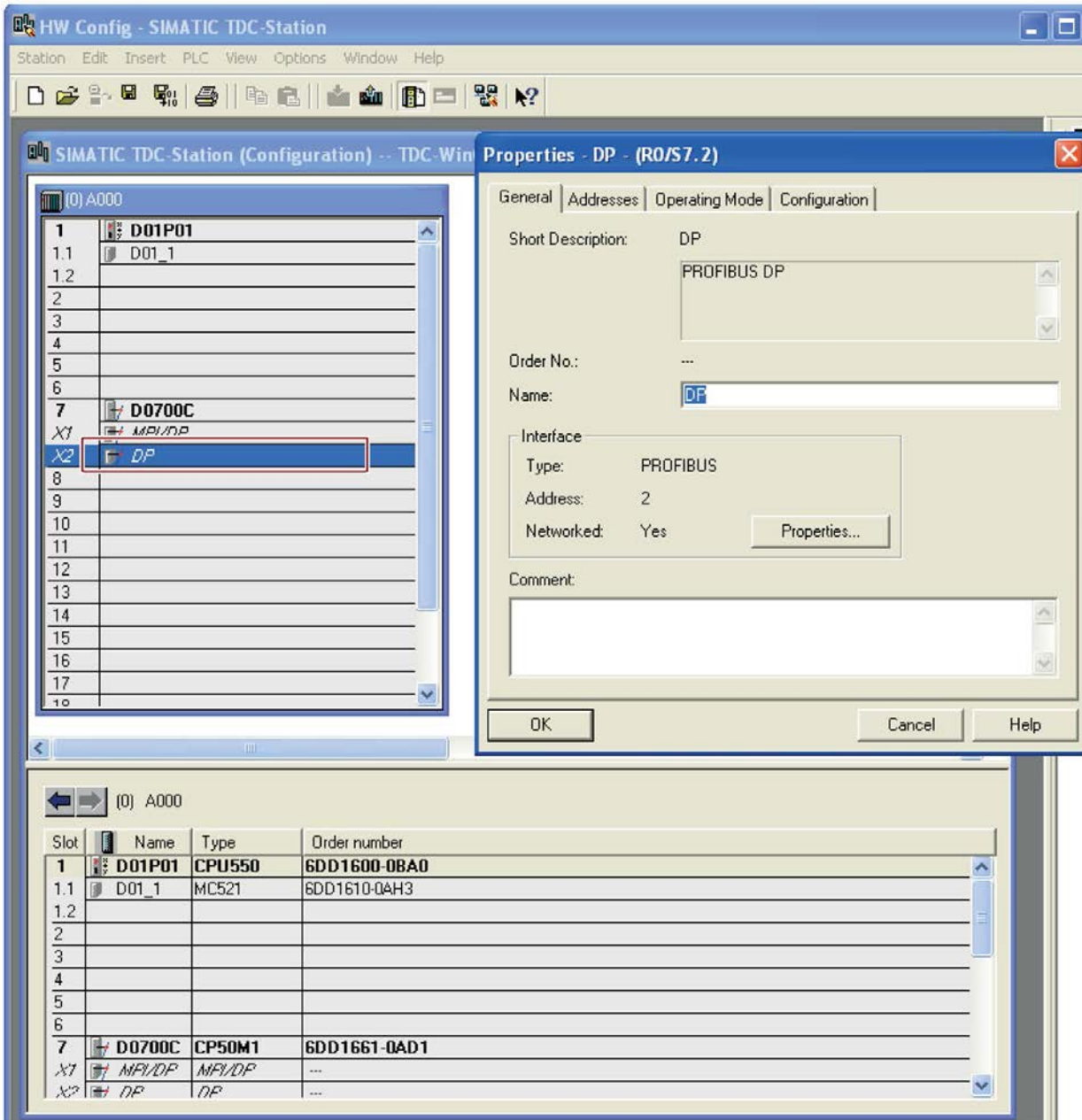


Figure 3-84 Open the Properties dialog

3. Click "New" in the next window to insert a new "Subnet".

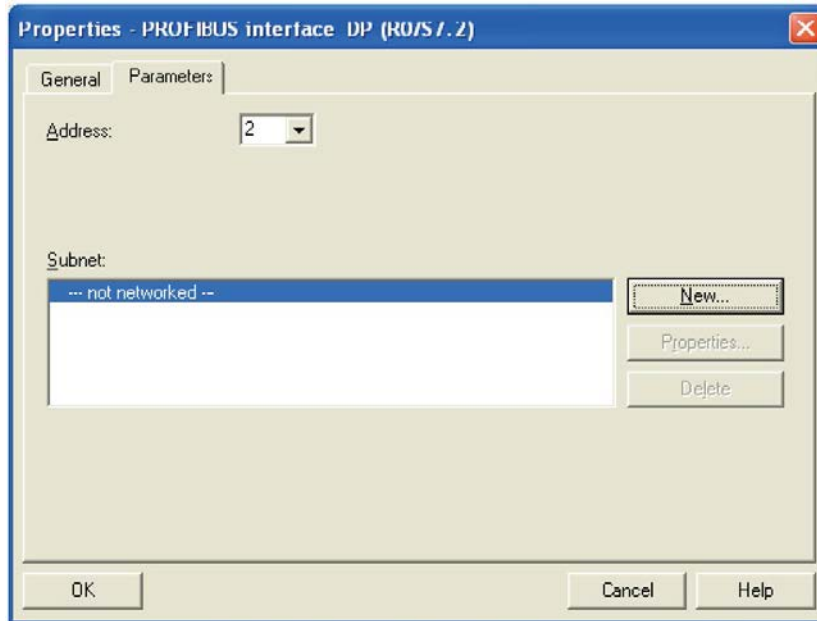


Figure 3-85 Create new subnet

4. You can rename the subnet and then select the "Open network settings..." tab.

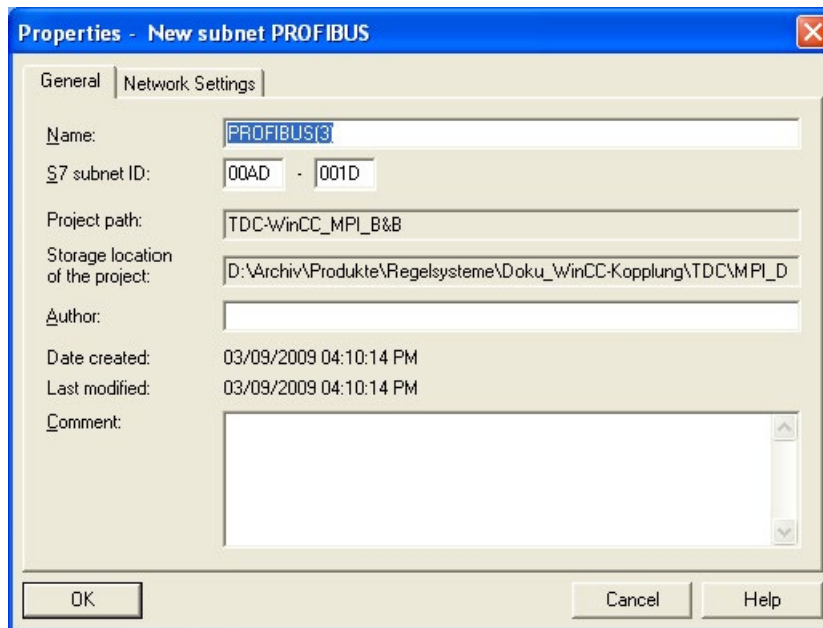


Figure 3-86 Specifying properties

5. Set the baud rate in the next window.

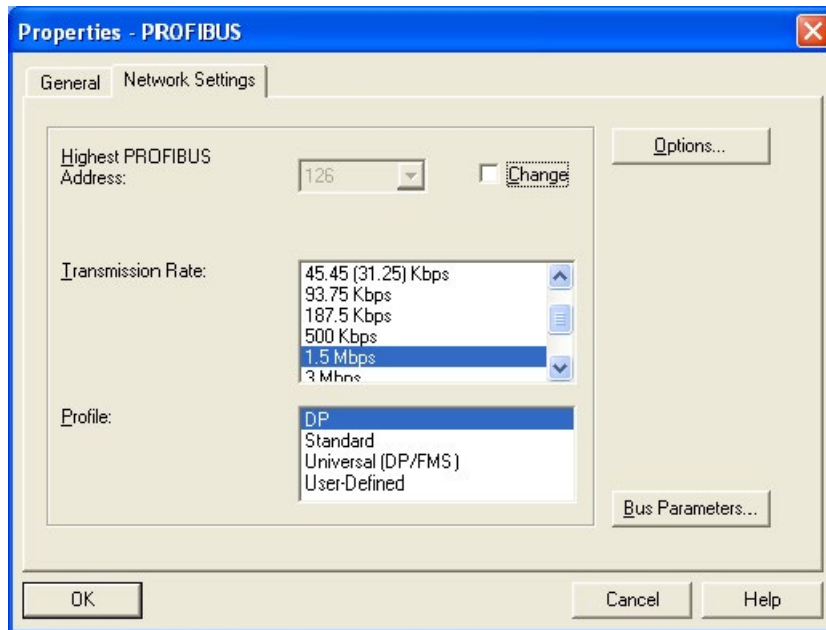


Figure 3-87 Set transmission speed

6. Close this window and set the address in the next window. Close all other windows with "OK".

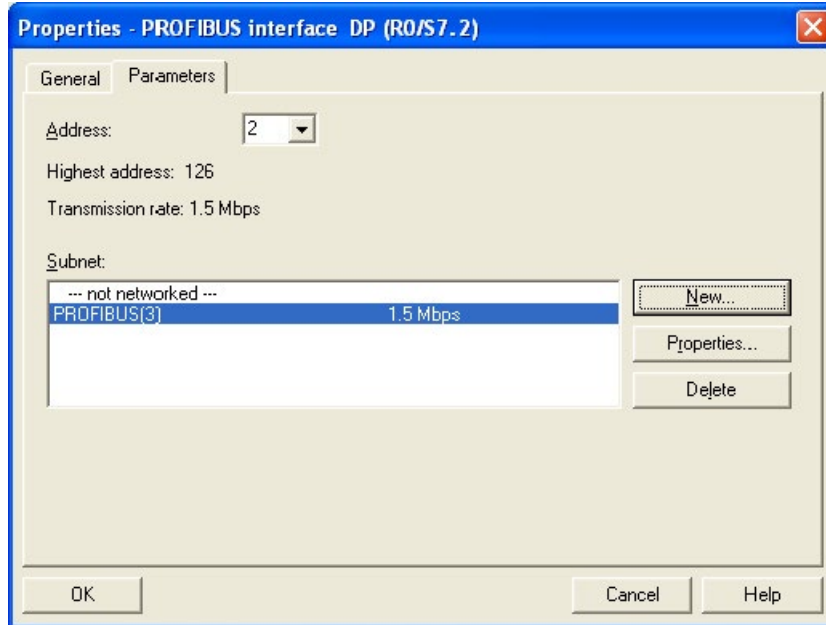


Figure 3-88 Set the address

7. You successfully configured the PROFIBUS DP or MPI segment as shown in the next figure.

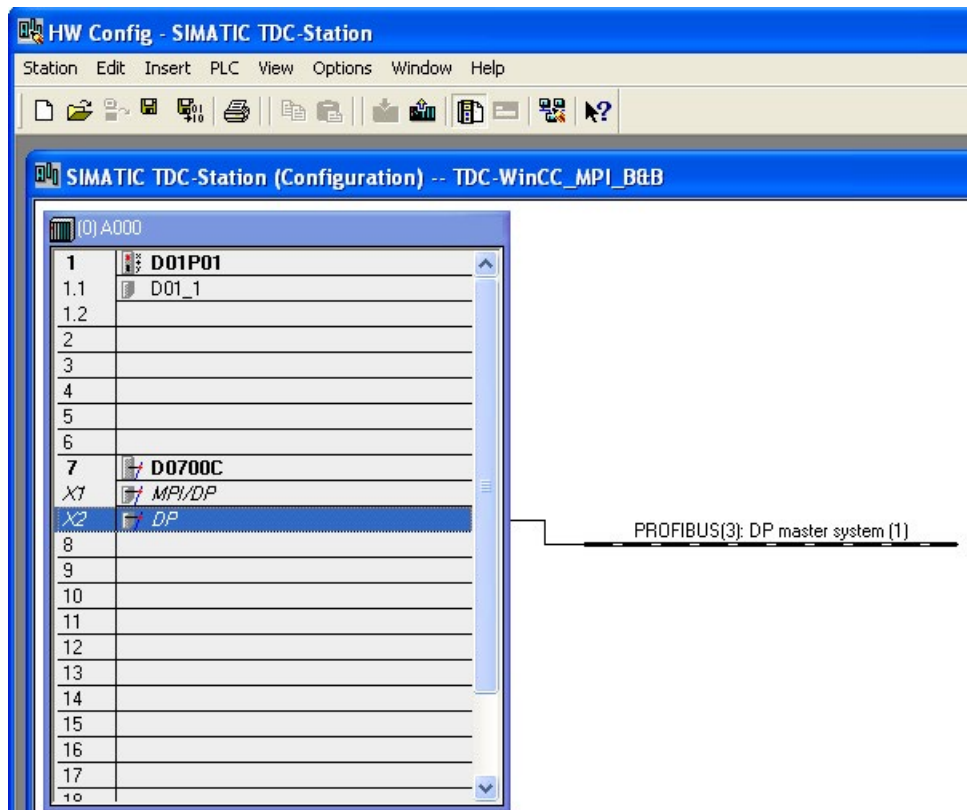


Figure 3-89 Result: Configured PROFIBUS line

Synchronize the station address and slot number between the TDC station and WinCC as follows:

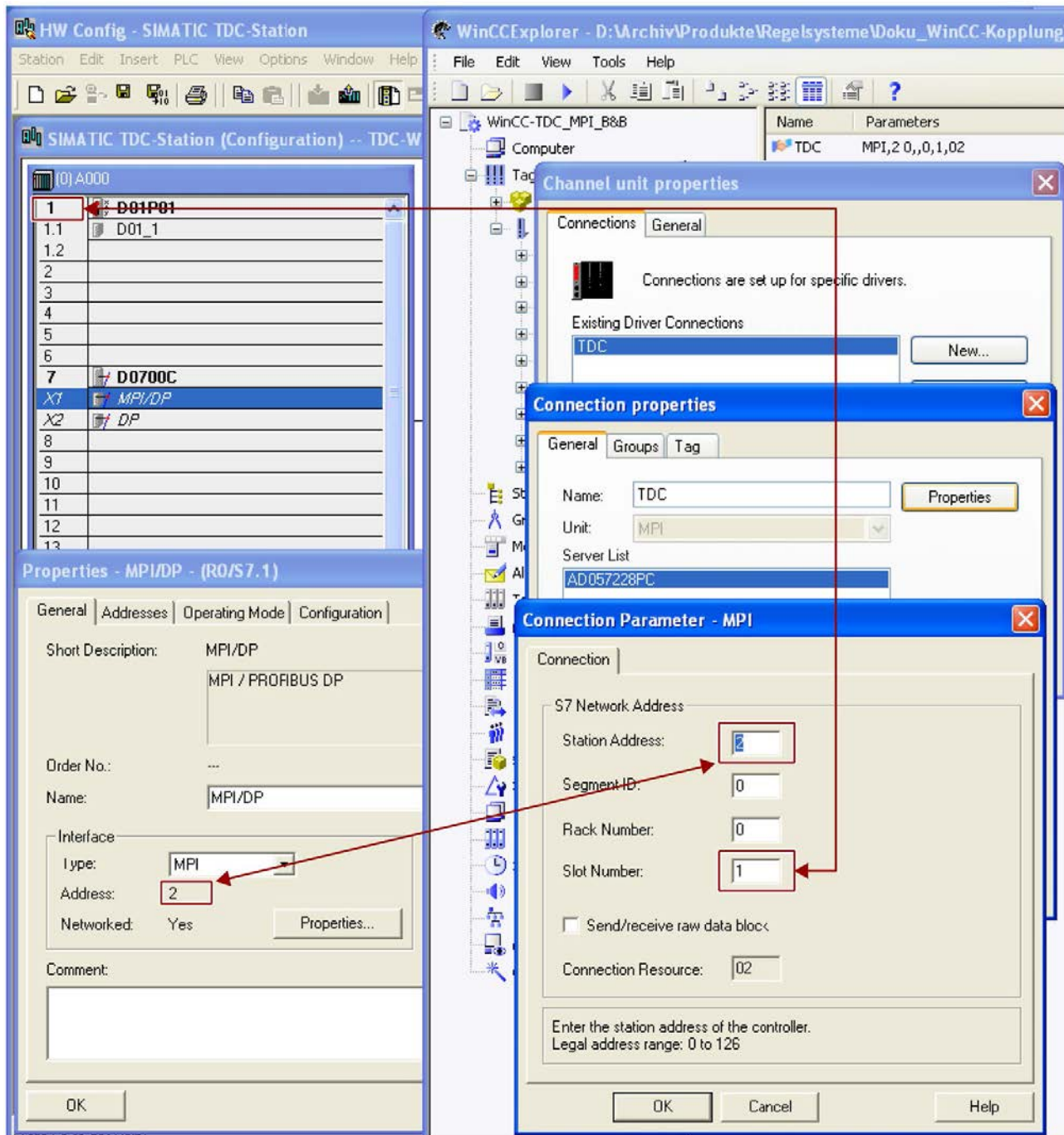
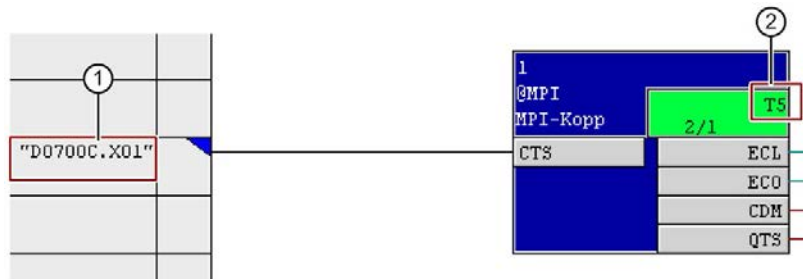


Figure 3-90 Synchronize station address and slot

3.14.3.2 CFC configuration

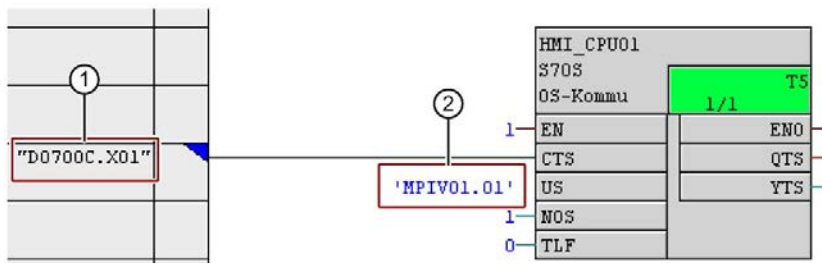
To initialize MPI or DP coupling, configure the corresponding central block "@MPI" or "@PRODP" in a task between 32 and 256 ms and interconnect the CTS connector of this block with the slot and front connector of CP50M1.



- ① "Name.Front connector" of the CP
- ② Tx: 32...256 ms

Figure 3-91 Configuring central block

In addition, configure the "S7OS" function block in a task between 32 and 256 ms and interconnect it as shown in the figure:



- ① "Name.FrontConnector" of the CP
- ② User-defined unique, 6-digit name.CPU slot number

Figure 3-92 Configure function block S7OS

3.14.3.3 WinCC configuration

1. Select **Tag management > right-click > Add new driver > SIMATIC S7 Protocol Suite.CHN > Open** to create a new driver.

Continue with the next step if the driver already exists.

2. Select **MPI > right-click > New connection** to create a new connection.

Likewise, select **PROFIBUS DP > right-click > New connection** in the same dialog to create a corresponding connection if you want to use PROFIBUS DP coupling.

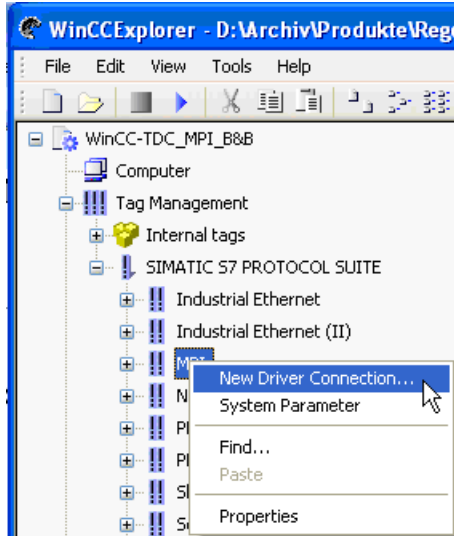


Figure 3-93 Create new connection

3.14 WinCC connection to SIMATIC TDC via standard channel (SIMATIC S7 Protocol Suite.CHN)

3. Assign a name to the connection in the dialog, press the Properties button and enter the connection parameters.
4. In the "Connection parameters" screen form of WinCC, copy the station address and slot number from the hardware configuration as shown in the following figure.

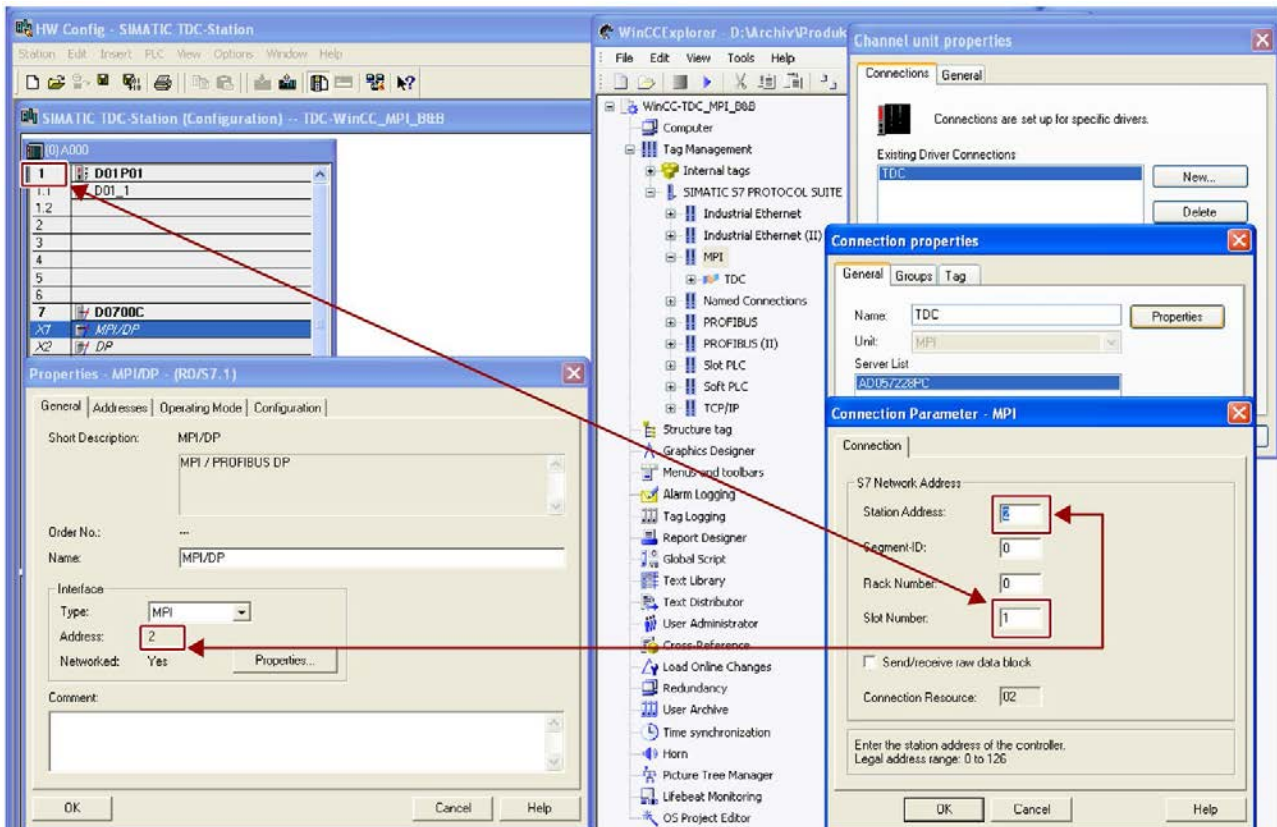


Figure 3-94 Set connection parameters

You can refer to the previous chapter for instructions on further configuration procedures, as these do not differ from the procedures for TCP/IP coupling.

Note

WinCC project transfer

When you transfer a WinCC project to another computer, you need to manually update the computer name in the computer properties and save the project *before mapping*. Otherwise, errors can occur in the project.

3.14.4 Configuration using the "D7-SYS-OS Engineering Tool"

The "D7-SYS Engineering Tool", referred to as "mapper" as of herewith, creates tags that are used for the selected connectors of the CFC function blocks and for further processing in WinCC.

The following chapter describes how to use this tool.

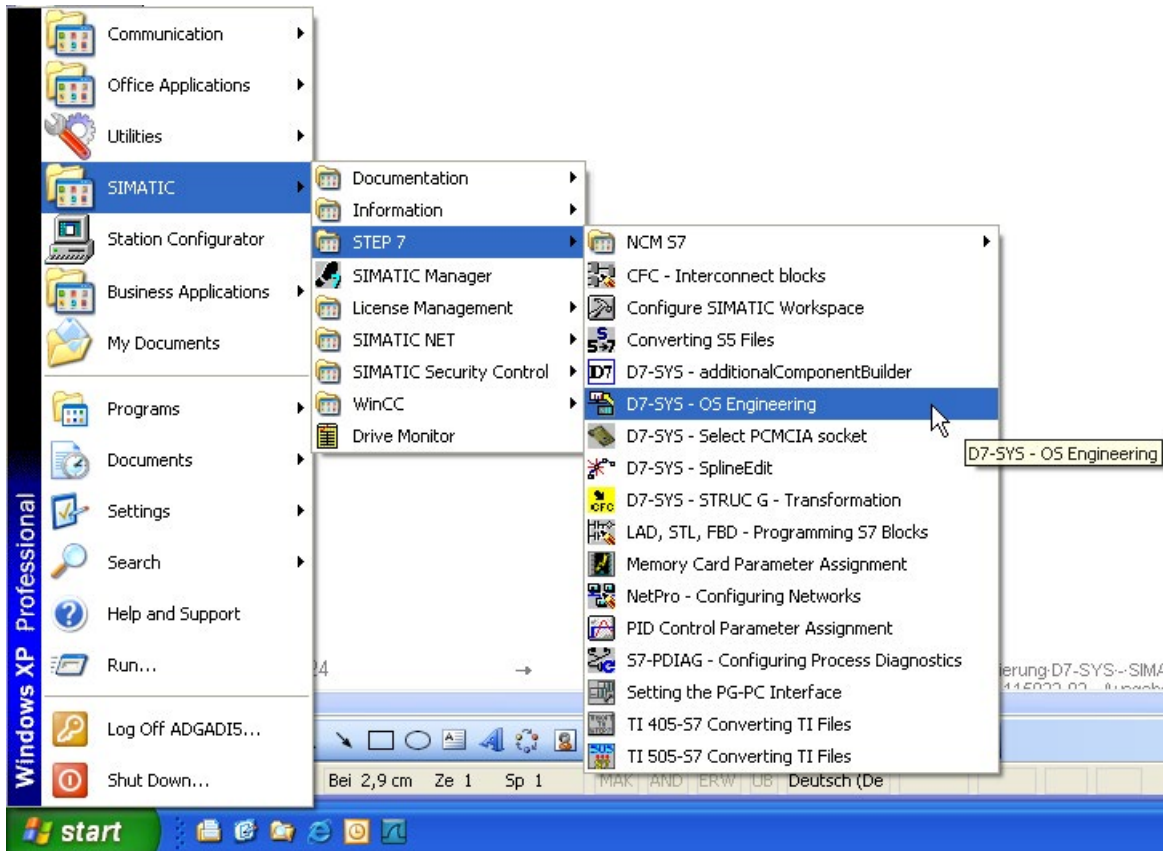


Figure 3-95 Open the mapper

3.14 WinCC connection to SIMATIC TDC via standard channel (SIMATIC S7 Protocol Suite.CHN)

Mapping requirements:

- Select the CFC function block connectors and then initiate a conclusive compilation with active "Create address book" option as described in the previous chapter.
- Inserted PC station with communication module and WinCC application:
- The "NetPro" configuration should be checked as to whether all stations are interconnected by means of the selected coupling type, e.g. over TCP/IP in this case. Proceed likewise for MPI or DP couplings.
- To enable data transfer to WinCC, you first need to start WinCC Explorer and load the corresponding WinCC project.
- The SIMATIC Manager must also be closed.

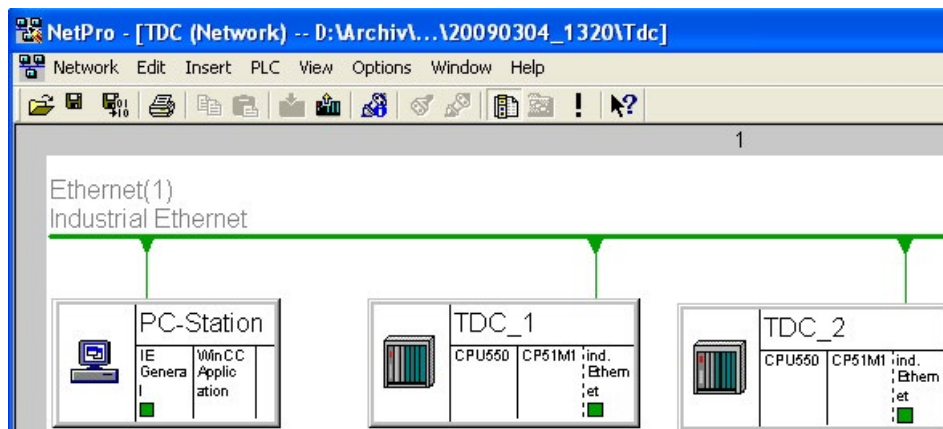


Figure 3-96 Check NetPro configuration

Following the call, click the "Open" icon on the left to select the target project.

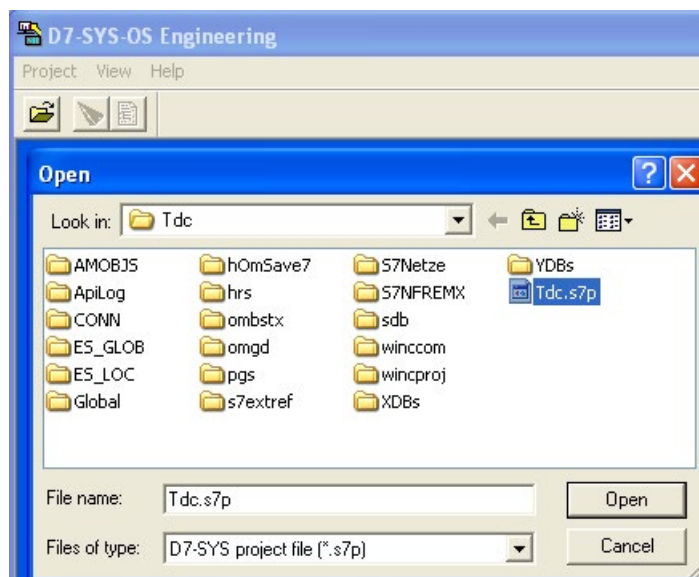


Figure 3-97 Open project

Double-click the second icon (Wizard's hat) to launch the Wizard:

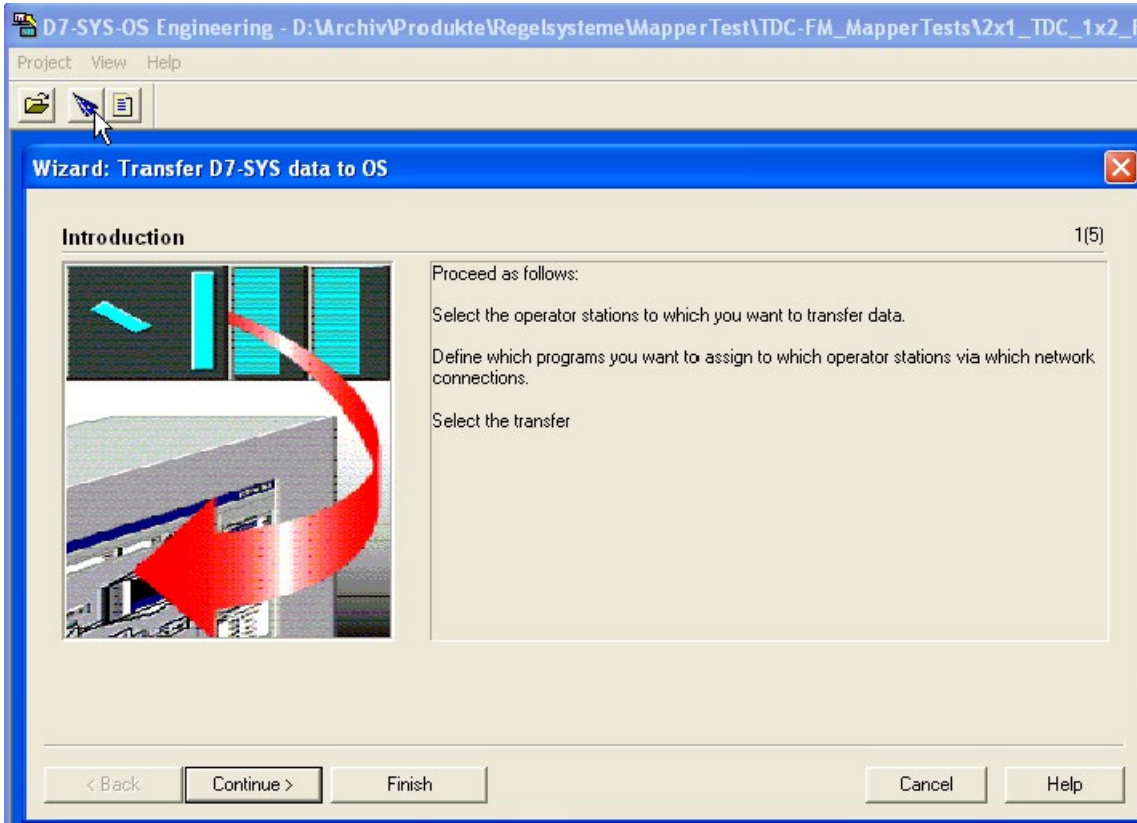


Figure 3-98 Start wizard

Click "Next":

Select the Operator Station in the next screen:

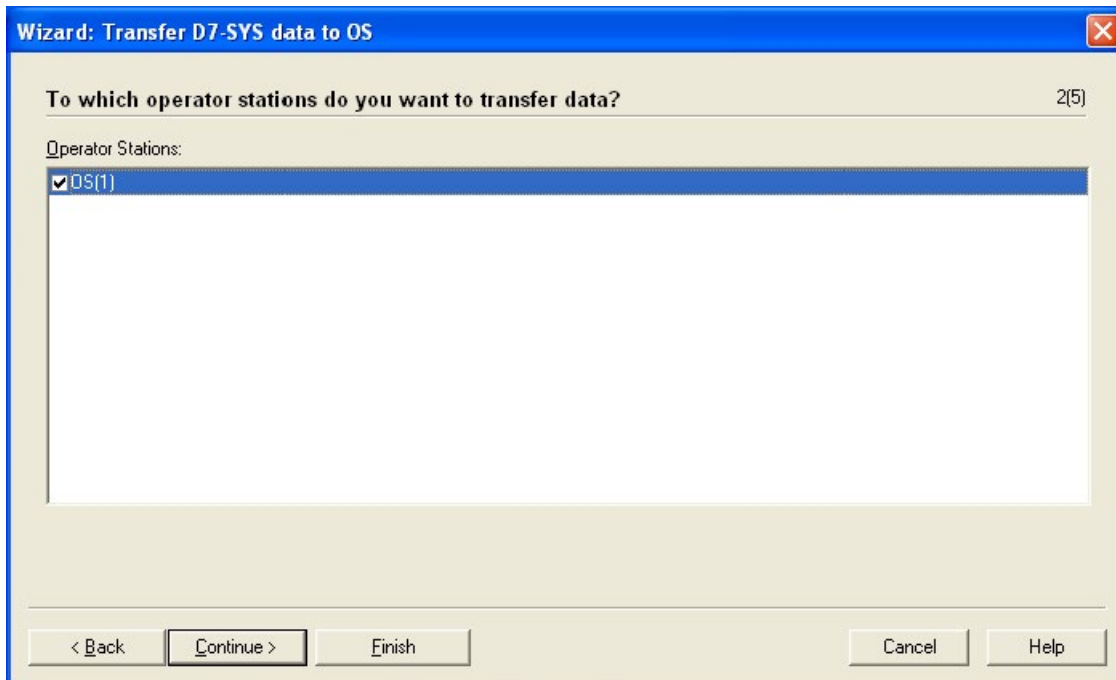


Figure 3-99 Select OS

Click "Next":

Select and assign the programs to the operator stations in the next step:

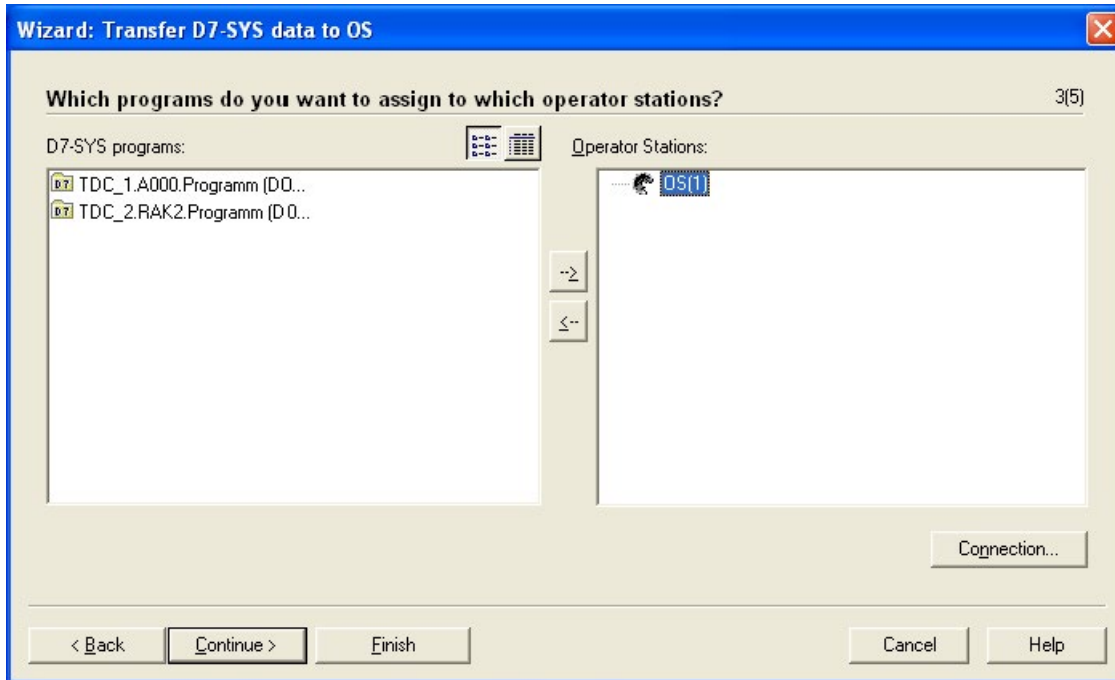


Figure 3-100 Assign programs of the OS

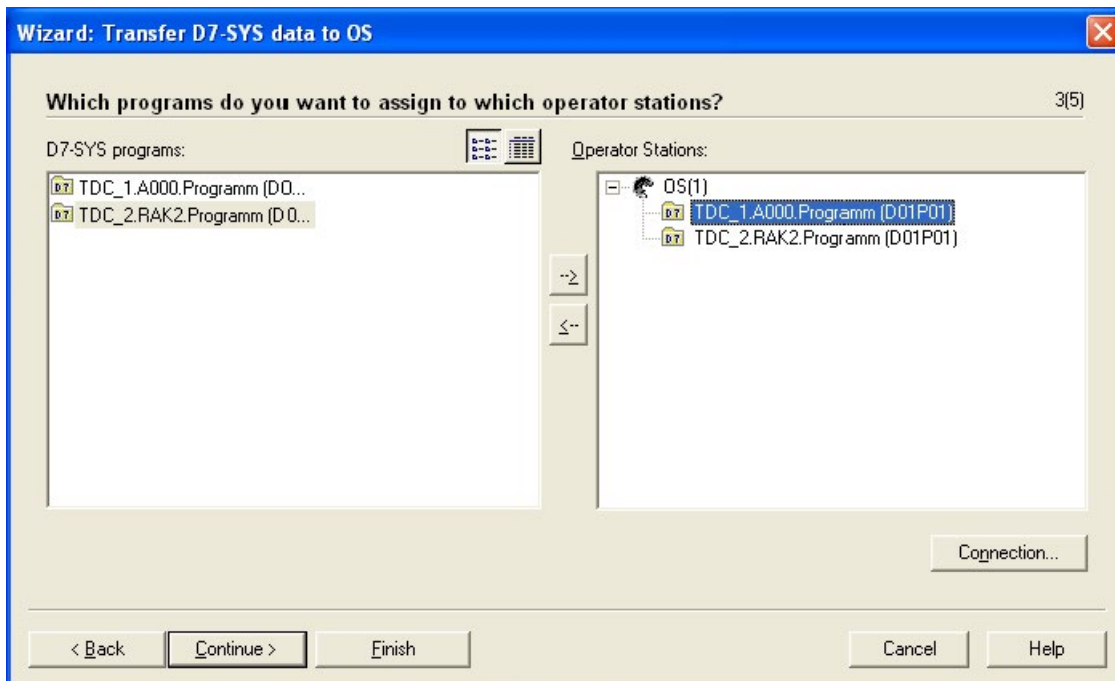


Figure 3-101 Display the selected programs

3.14 WinCC connection to SIMATIC TDC via standard channel (SIMATIC S7 Protocol Suite.CHN)

Click "Connection" if you want to verify the connection parameters. This action is not necessary if the configuration in "NetPro" is properly completed. However, if different couplings are used (TCP/IP, MPI or DP) you must select the respective connection in this dialog.

The settings are not saved.

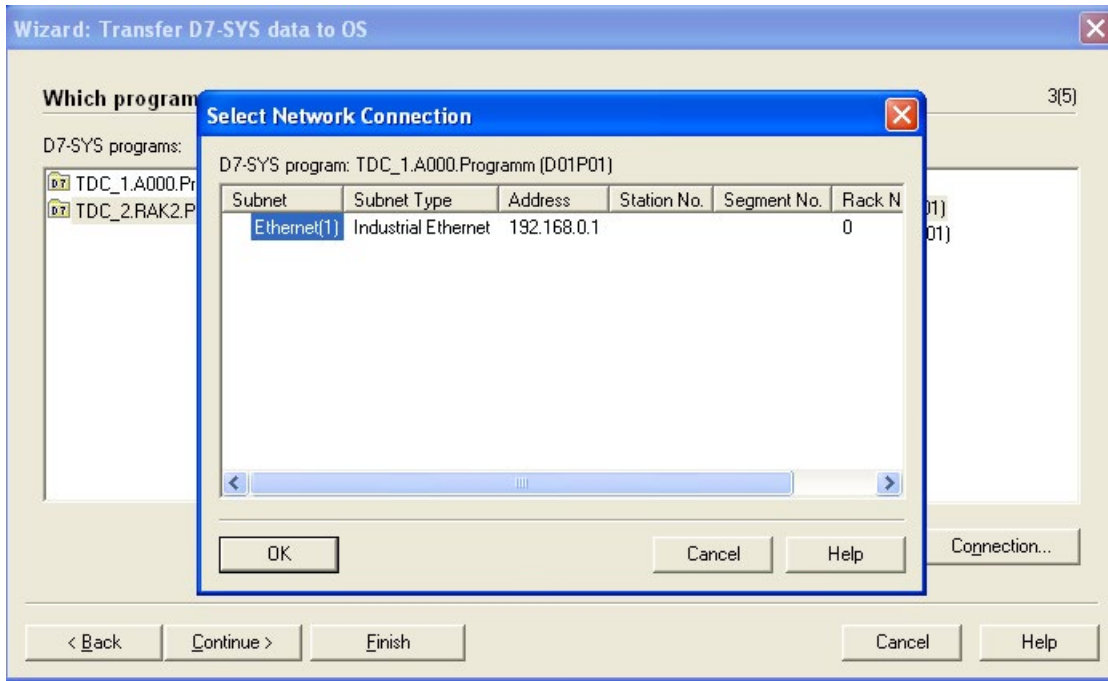


Figure 3-102 Check connection parameters

Click "Next":

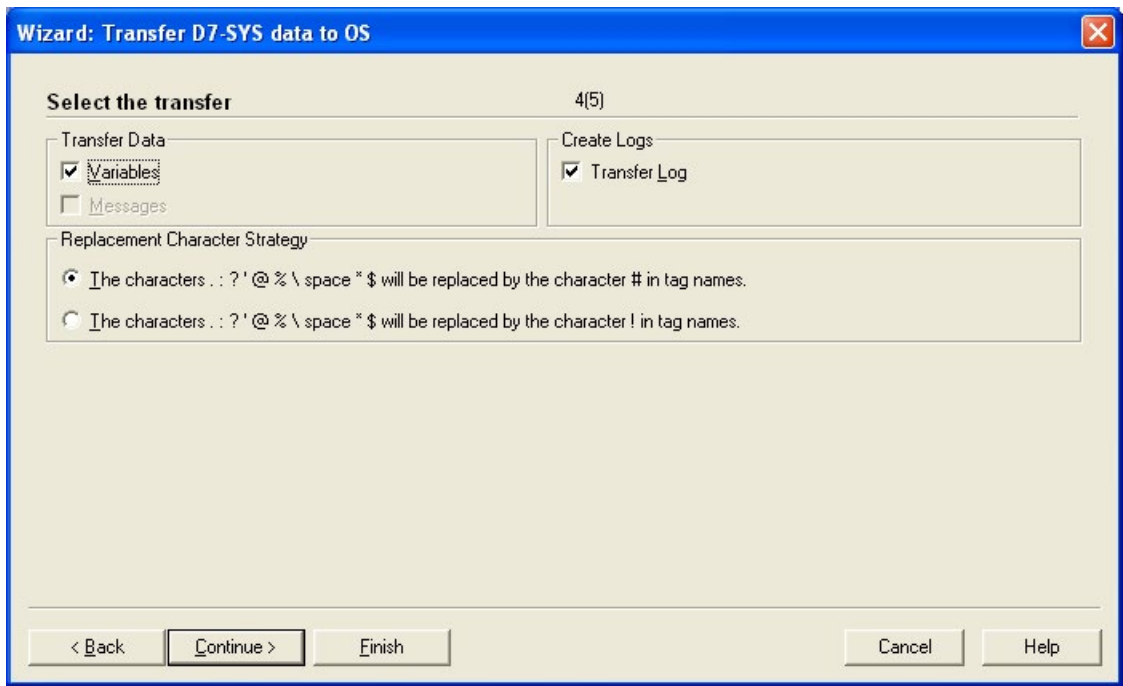


Figure 3-103 Transfer data

Click "Next":

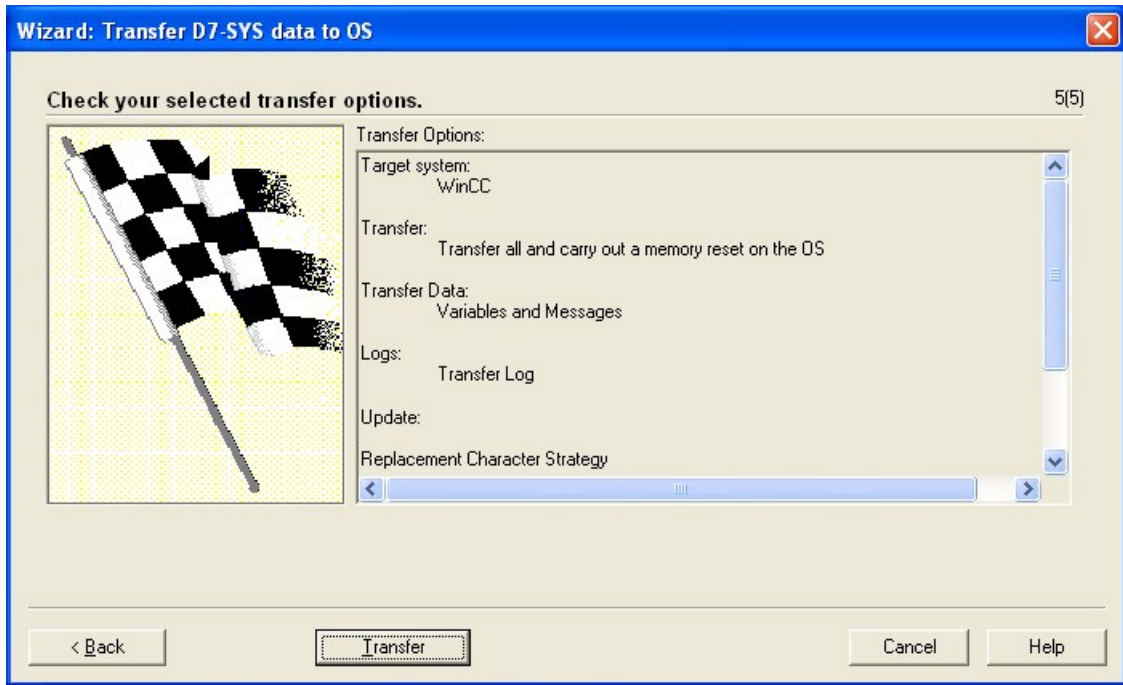


Figure 3-104 Check transfer options

Click "Transfer" to launch the actual mapping process:

Data transfer (mapping) is now busy:

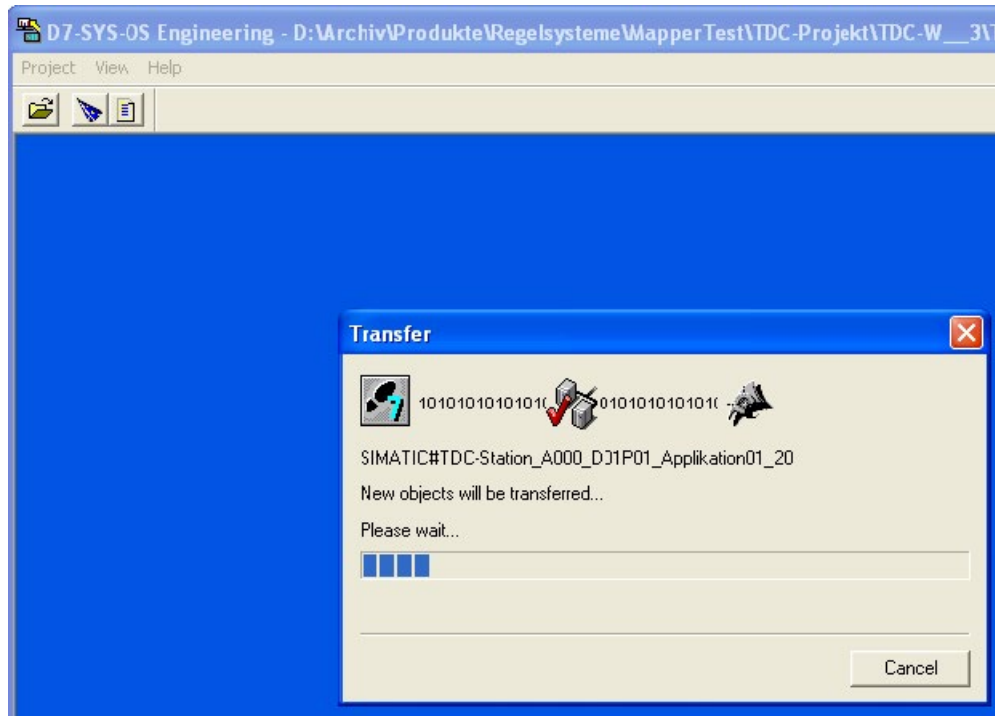


Figure 3-105 Transfer in progress

Data transfer is now completed:

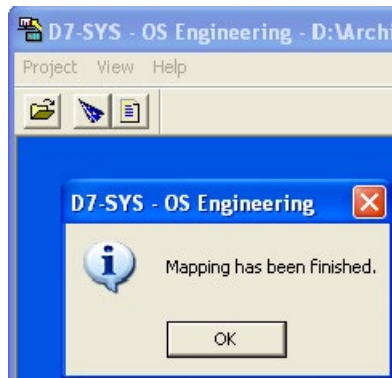


Figure 3-106 End mapping procedure

Click "OK" to close this dialog.

The tags you created are now listed in the tree structure of WinCC Explorer:

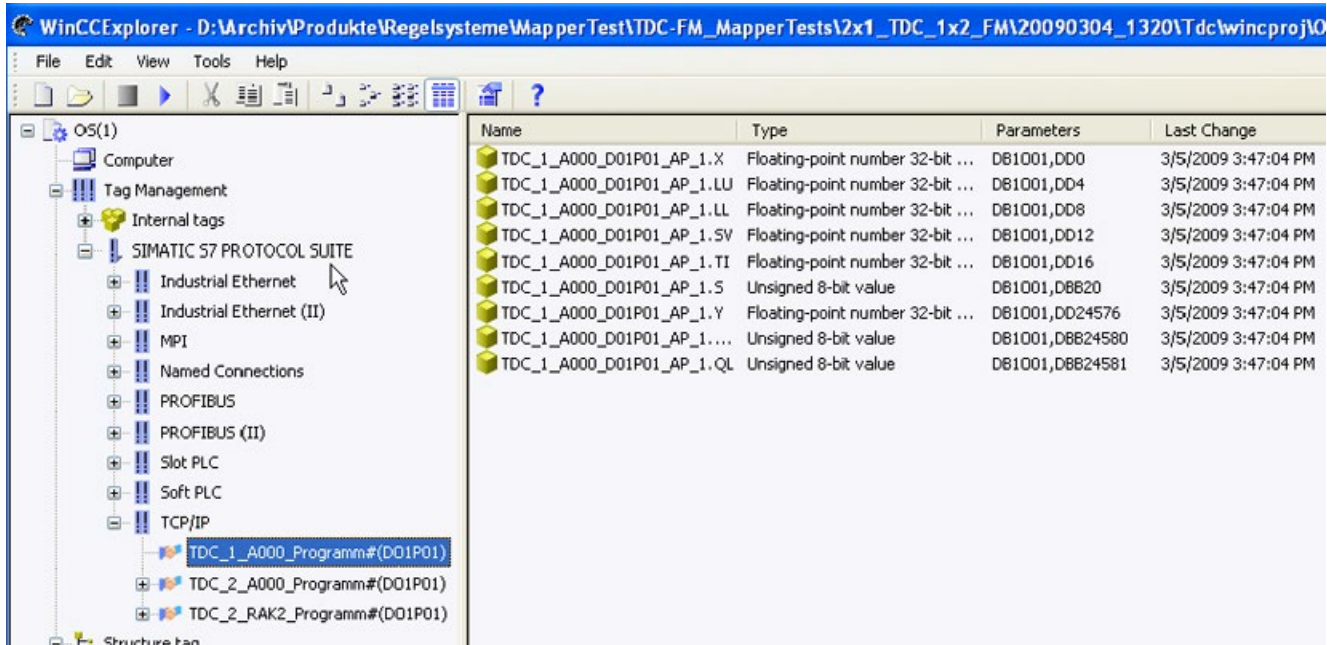


Figure 3-107 Display of the created tags

You still have to check the system parameters of the coupling shown in the following figures:

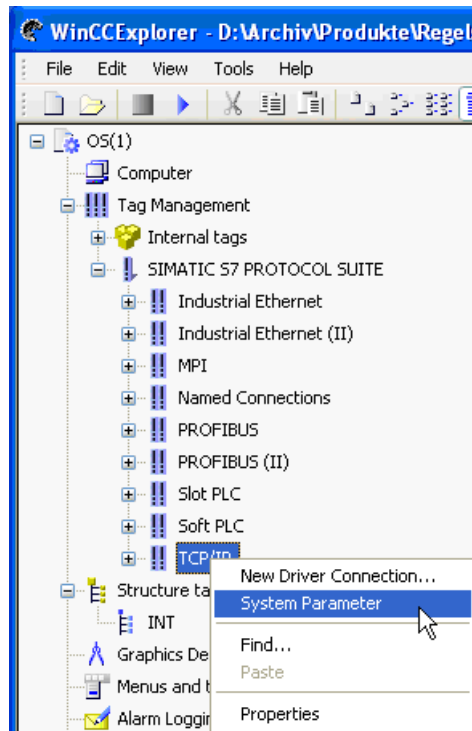


Figure 3-108 Call system parameters

Disable cycle management by the automation system

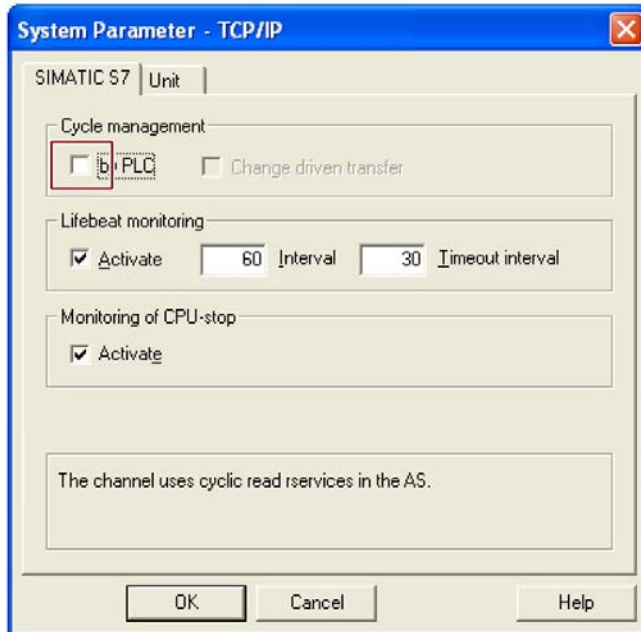


Figure 3-109 Disable cycle management

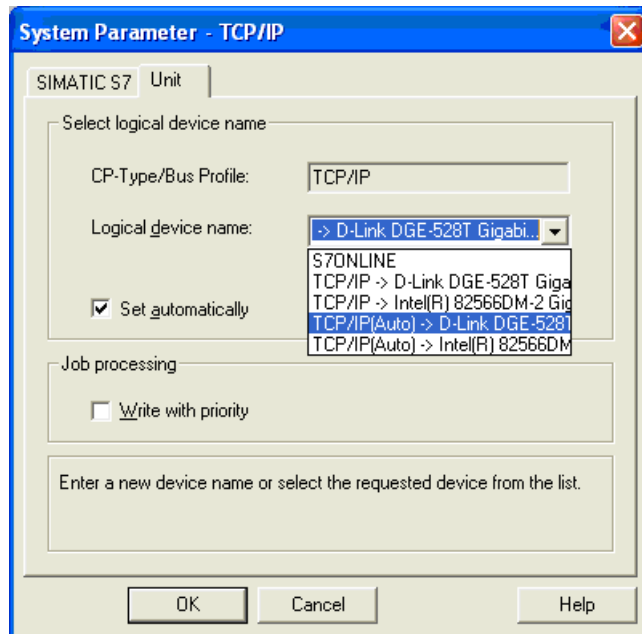


Figure 3-110 Select device name

Restart WinCC to activate the changes to system parameters. You can then access the configured tags.

3.15 Communication with WinCC (TCP/IP)

Introduction

Based on a simple configuration sample, this User Manual shows how you can integrate WinCC in SIMATIC TDC via TCP/IP coupling. The document describes all necessary configuration tasks (including hardware and software requirements). It does not cover the handling of the necessary software tools and, instead, provides references to the relevant User Manuals.

Note

Increased communication load

Depending on settings, PG/PC tools such as WinCC or CFC online and HMI Panels create an increased communication load.

This can result in longer response times and communication failures.

3.15.1 Requirements

Software

SIMATIC D7-SYS as of version 8.1

Hardware

PC configuration station: Network adapter for TCP/IP, e.g. 3COM

SIMATIC TDC hardware configuration

Table 3- 27 Hardware configuration for the example configuration

System:	D7-SYS V5.1 or higher	
Rack:	UR5213	21 slots with fan
Slot 1:	CPU551	CPU (with local service interface)
Slot 1.1:	MC500	4 MB Flash memory module
Slot 2:	CP50M1	Buffer memory module
Slot 18:	CP51M1	TCP/IP interface

3.15.2 Process tags

A SIMATIC TDC station must be configured and parameterized and a test chart generated using the CFC configuration tool. A description of the hardware configuration is available in section SIMATIC TDC - Hardware configuration. Section CFC configuration (Page 242) describes how to create a configuration with the CFC.

SIMATIC TDC configuration

Although the CFC for WinCC coupling does not have to be created in a separate chart, this procedure is recommended for the sake of transparency. You need process tags for the following function blocks for coupling SIMATIC TDC and WinCC:

CPU	Required function blocks
CPU551	<ul style="list-style-type: none"> • @TCPIP - TCP/IP central block coupling • SER - Service block
CPU555	No function blocks need to be configured.

The blocks are wired as follows:

(the description covers only the relevant I/Os)

FB @TCPIP

I/O name	Meaning	Example
CTS	Name of the interface used (CP51M1)	'D1800C'

FB SER

I/O name	Meaning	Example
CTS	Coupling module name	D1800C.X01
US	Address parameters	'Ser01.01'
NPG	Number of PGs	2
NOS	Number of OS	0
TLF	Message frame format	0

Note

In addition, the @GLOB central block must be configured for the buffer memory module and the @TCPIP central coupling block (in the case above, also the @LOCAL, as all of the connections are connected to D01P01).

FB @GLOB

I/O name	Meaning	Example
CTS	Buffer memory module name	'D0200A'
CDV	Memory re-allocation (1)	0

WinCC configuration

For the particular configuration sample, it is sufficient to use a basic WinCC configuration with a small number of I/O fields.

This section does not provide details related to WinCC configuration. If you need more information, refer to the comprehensive WinCC Configuration Manuals. We recommend the SIMATIC WinCC, Getting Started Manual to get you started with a first WinCC configuration.

3.15.3 Bit messaging

SIMATIC TDC configuration

It is not necessary to create a configuration for process value monitoring for bit messaging in WinCC. The specific tag bit initiating a specific message is always selected in WinCC. The configuration rules for the output of process tags remain valid.

WinCC configuration

In addition to the configuration of process tags, you need to create an ALARM logging configuration. The WinCC configuration is not covered in this section. For more information, refer to the comprehensive WinCC Configuration Manuals. We recommend the SIMATIC WinCC, Getting Started Manual to get you started with a WinCC configuration containing a message configuration.

See also

Communication utility alarm logging (Page 194)

3.15.4 SIMATIC TDC messages

Note

SIMATIC TDC PMC TCP/IP COMMUNICATION CHANNEL

This section refers to the use of the SIMATIC TDC PMC TCP/IP COMMUNICATION CHANNEL product.

The SIMATIC TDC PMC TCP/IP COMMUNICATION CHANNEL product is being discontinued and the SIMATIC TDC messaging procedure is therefore not supported by the local PN interface of the CPU555 (use an Ethernet CP for this).

SIMATIC TDC configuration

To output messages from SIMATIC TDC to WinCC, you need the WinCC block MM in addition to the configuration of the process value output:

MM - Message Manager The blocks are wired as follows (only the relevant I/Os are described):

FB MM

I/O name	Significance	Example
CTS	Processor name	D01P01
AR	Channel name (identical with the AT I/O of the MSI block)	RECCHAN
NZ	Number of cycles per transmission	5
NL	Number of connected LI blocks	1
MEM	Diagnostics triplet	0
TGL	Diagnostics triplet	0
CVP	Connection with LI, CCV I/O	>LI.CCV

Note

You also need to configure the central message block @MSI, message output block MSI, and message block MERF0 or another MERFxx.

FB @MSI

I/O name	Significance	Example
CMS	Name of message system	MYMELD
CMT	Message text (not output)	""
NOM	Number of messages that can be saved	200
SAV	Message buffer in buffered RAM	0
RP	Prefix for communication errors	0
MUN	Enable for message entries	1

FB MSI

I/O name	Significance	Example
CMS	Name of message system	MYMELD
CTS	Coupling module name	D01P01
AT	Address parameters	RECCHAN
RP	Prefix for overflow messages	0
SNV	Message number output	1
STM	Message text output	0
STC	Message text output with constant length	1
SSF	Output format	1
EN	Enable	1
MUN	Enable for message entries	1

FB MERFO

I/O name	Significance	Example
CMS	Name of message system	MYMELD
MT	Message type	1
RP	Prefix	0
RS1	Suffix of incoming message	10001
RS2	Suffix of outgoing message	00005
EN	Message enable	1
IS1	Message trigger	16#0
SM	Save message	0

WinCC configuration

In addition to the configuration of process tags, you need to create an ALARM logging configuration. The WinCC configuration is not covered in this section. For more information, refer to the comprehensive WinCC Configuration Manuals. We recommend the "SIMATIC WinCC, Getting Started" Manual to get you started with a WinCC configuration containing messages. The assignment of SIMATIC TDC message numbers at the message blocks (RS* connections) to the message numbers generated in WinCC can only be identified by the "PMC message no." that is generated based on the message numbers of the signal list.

3.15.5 Address book generation in CFC editor

To generate the signal list for WinCC, ADRIMP needs the symbol information from the SIMATIC TDC processors. SIMATIC TDC generates an ASCII file that contains this information for all CPUs. The file name consists of the rack names and CPU number, separated by an underscore "_" and the file name extension ".ADR".

To generate the address book, select **Options - Customize compilation....** to open the relevant project chart. Select the Generate address book check box and confirm with OK. Select the **Chart and Compile** menu commands. The address book will be created during compilation. Select **Options - Logs** to search for the path to the generated address book.

Note

OCM must be activated for all SIMATIC TDC I/Os to be accessed in WinCC (see manual "D7-SYS - STEP 7, configuring CFCs and SFCs (<http://support.automation.siemens.com/WW/view/de/8776786/0/en>)").

3.15.6 Communication setup SIMATIC TDC <-> WinCC

Activating WinCC

To establish communication between SIMATIC TDC and WinCC, the import data of the WinCC database must be assigned to the I/O fields of the graphic configuration. For this purpose, select the corresponding fields in Graphics Designer and click in the interactive configuration dialog. Each field can be assigned one of the imported tags. On completion of this assignment, select the File menu from the main menu to save the data. Specify the connection properties prior to the start of runtime.

In the Control Center

- Click on Tag Management
- Click on TDC PMC TCP
- Right-click TCP/IP Unit 1
- Click on Properties
- Click on the Properties < Channel Unit (to configure PMC parameters)

Click on the connection. Enter the TCP/IP addresses (local/remote) and the port number (local/remote)

- Confirm with OK
- Select "File" in the Control Center
- Click Activate

WinCC is now ready to exchange data with SIMATIC TDC

Activating SIMATIC TDC

Switch on the configured racks. The connection is up between SIMATIC TDC and WinCC on completion of the rack startup. Data is now exchanged in cyclic mode between SIMATIC TDC and WinCC.

3.16 Communication utility Trace

Three different services are available for trace value recordings:

- **Simple Trace**

Contains only one output interface.

- **System trace**

Contains an interactive interface (request and response interface). To be able to use the system trace, you must develop your *own* tool that is capable of handling the request/response code of system trace. For this reason, this chapter will not cover the system trace in detail.

- **Analog Trace**

Is not provided a data interface. This service is always executed with the TRCC, TRCC_D and TRCC_I function blocks. A corresponding description is not available in these configuration instructions.

3.16.1 Simple Trace

Simple trace allows you to log, save and output values of local processor connectors. Simple Trace consists of a control block @TCP and one or several acquisition blocks TRP, TRP_B, TRP_D and TRP_I or TRHI. You may configure any number of acquisition blocks with custom sampling times.

3.16.1.1 Mode of operation of @TCP

The @TCP block is essentially assigned the following tasks:

- Control of the trace value acquisition blocks TRP, TRP_B, TRP_D and TRP_I.
- Output of logged values.
- Troubleshooting

The following paragraphs describe these tasks in detail.

Controlling trace value acquisition

Trace value logging and output are controlled by the input connectors STA and TBR of the @TCP.

Logging trace values

Logging by the acquisition blocks is initiated as follows:

- R connector: 0
- TBR connector: 0
- STA connector: 0 to 1 transition

Logging is active until a 1 to 0 transition is detected at the STA connector (premature cancellation of trace value logging by RESET is not covered in this chapter). The @TCP block is in *logging mode* while logging is active. Trace value logging is terminated if a transition from 1 to 0 is detected at the STA connector. Trace value logging is terminated immediately if the 0 value is set at input connector TDC. Otherwise, the @TCP FB waits for the number of cycles defined at the TDC before it terminates trace value logging.

Output of trace values

After trace value logging has terminated, you can initiate the output of trace values at the @TCP block as follows:

- R connector: 0
- TBR connector: 0 to 1 transition
- STA connector: 0
- OUT connector: Group frame output (no/yes)
- CID connector: If group frame output is not set:

ID code of the connector of which the logged values are to be output.

The @TCP block is set to the so called *output state* while data output is active (a temporary status change to the wait state is possible, as long as the data interface is temporarily not accessible).

Two options are available for the output of trace values:

- The trace values of all acquisition blocks are transmitted in single frame. This is the case if connector OUT is set to the value 1.
- Use the CID connector to select the acquisition block of which the trace values are to be output. The logged trace values are output in one or several response frames via the data interface. If it is not possible to transmit the logged values of a connector in a single response frame, the data is automatically segmented and transmitted in several response frames. The logical sequence of the frame packets is specified based on two counters in the frame: The counter *number of packets* defines the number of data packets that are necessary to output all values of the relevant connector and the counter packet number defines the number of the current packet (the numbering convention for both counters is specified in the next chapter). Data output is terminated after all trace values have been output, or at the transition from 1 to 0 at the TBR connector.

Resetting the Simple Trace

The Simple Trace function can be reset by setting input connector R to 1. This action immediately stops any active logging process, or the output of trace values. Moreover, the trace buffer will be reformatted, i.e. all values in the buffer will be lost. The reset command remains active until the R connector is reset to 0.

Troubleshooting

When communication errors occur, a corresponding entry is made in the communication error field and the @TCP block is deactivated (QTS connector = 0). The corresponding communication error number is additionally set at the YTS connector.

Note

For information on the cause of the error as well as troubleshooting instructions, refer to the "D7-SYS - Error codes for fast access" document.

3.16.1.2 Mode of operation of the acquisition blocks

The TRP, TRP_B, TRP_D and TRP_I acquisition blocks are assigned the following tasks:

- Logging of specified input variables (trace logging)
- Troubleshooting

The mode of operation of all acquisition blocks is identical. They differ only with regard to the format of input variables to be recorded (TRP = Real, TRP_B = BOOL, TRP_D = Double Integer, TRP_I = Integer).

Trace logging

Each acquisition block logs the values of its input connector and saves these to a ring buffer. Each acquisition block is assigned one ring buffer. The ring buffers are assigned at the start of the cyclic phase. The entire trace buffer having a size that is determined by the value at connector TBL of the @TCP FB is distributed to the ring buffers. Each ring buffer is set to store the same number of values.

The acquisition blocks log exactly one value per cycle, provided trace value acquisition was activated by the @TCP central block. If connector values are logged by acquisition blocks that are configured for operation based at interrupt levels (non-cyclic logging), the acquisition time that consists of the date and time is included in the trace value entry in the trace buffer. The acquisition blocks do not record trace values if the @TCP block has deactivated trace value logging. Once the SSF connector of the @TCP is set to 1, the trace values that have been logged is recorded by the acquisition blocks in standardized floating point number format (Real values).

Troubleshooting

On detection of a communication error, a corresponding entry is made in the communication error field and the respective acquisition block is deactivated (QTS connector = 0).

In addition, the communication error number is output at the YTS connector.

Note

Deactivation of a TRP, TRP_B, TRP_D, or TRP_I as a result of a communication error has no influence on the other Simple Trace blocks, provided at least one other configured and functional acquisition block is available. For information on the cause of the error as well as troubleshooting instructions, refer to the "D7-SYS - Error codes for fast access" document.

3.16.1.3 Mode of operation of header block TRHI

The header block provides the parameters for group frames. A group frame is generated if the value at connector OUT of the @TCP FB is set to 1.

Header block TRHI only affects a trace system on the condition that a group frame output has been configured.

A TRHI FB configuration has no effect if no group frame has been configured.

Note

ATRHI FB configuration only affects the structure of a group frame. Refer to the "Group frames" *section*.

3.16.1.4 Simple Trace configuration

You may configure a single Simple Trace (@TCP=1), or multiple instances of this trace (@TCP>1).

Configuring exactly one Simple Trace

Exactly one control block @TCP and at least one acquisition block TRP, TRP_B, TRP_D or TRP_I must be configured for exactly one simple trace. The following configuration rules are binding:

- @TCP and acquisition blocks of a Simple Trace are assigned the same ID at the TRC connector.
- The value at the TBL connector of @TCP must be greater than zero.
- All acquisition blocks are assigned unambiguous ID codes at the CID connector.
- The acquisition blocks can be configured for operation with different sampling times.
- At least one acquisition block (any number are permitted) must be configured.
- It is possible to configure up to 255 acquisition blocks for group frame output (connector OUT of the @TCP FB = 1).

Example: Assignment of the initialization connectors for a configuration consisting of exactly one Simple Trace, four acquisition blocks and group frame output.

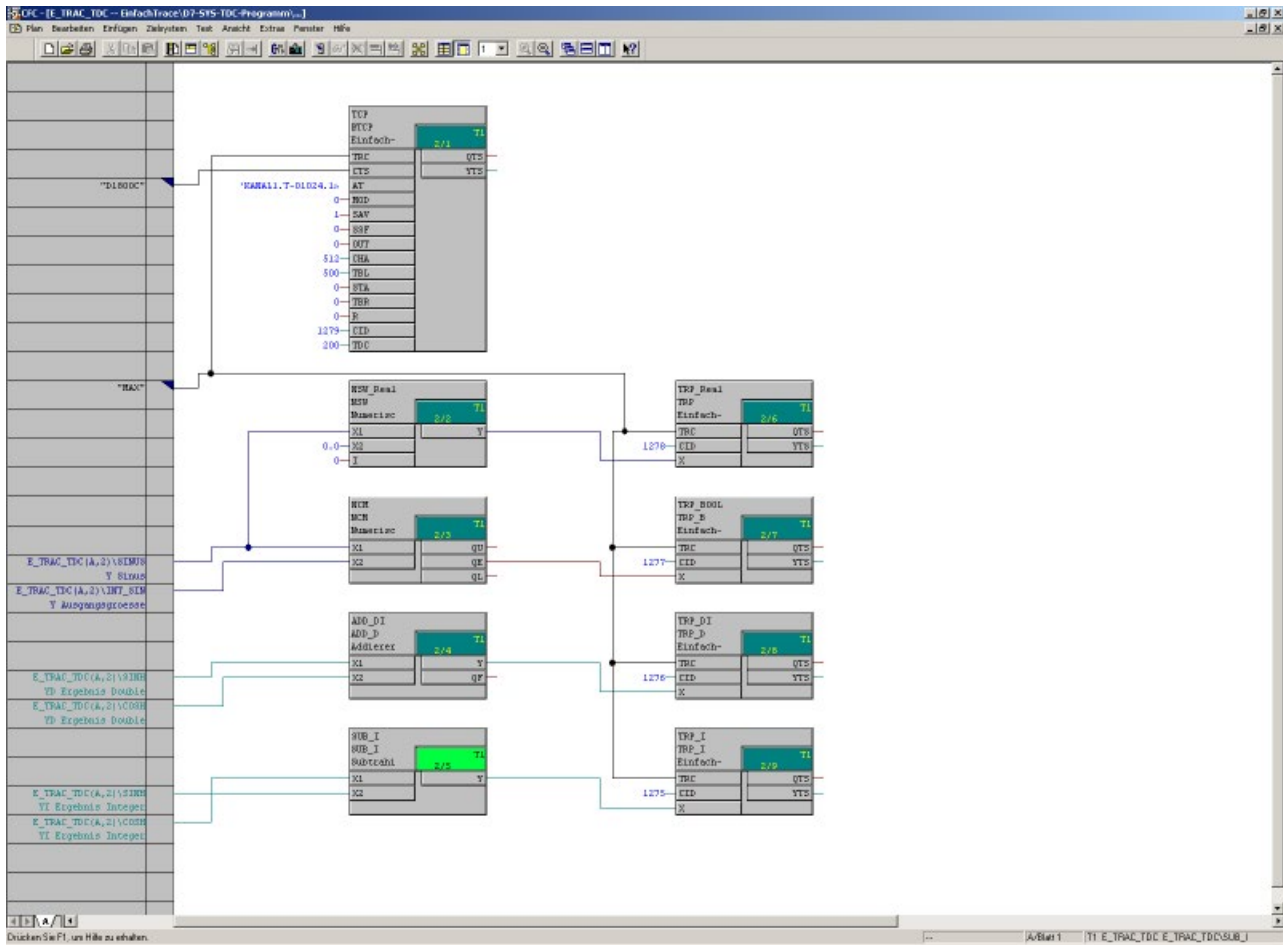


Figure 3-111 Example of a Simple Trace configuration

Configuring several Simple Traces

Along with the configuration of a single Simple Trace, it is possible to configure several Simple Traces in parallel.

A Simple Trace is formed exactly by the blocks that are assigned identical ID codes at the TRC connector. The following rule applies in addition to the configuration rules mentioned above:

All configured @TCP blocks must be assigned unique TRC ID codes.

Example: Assignment of the initialization connectors for the configuration of two Simple Traces.

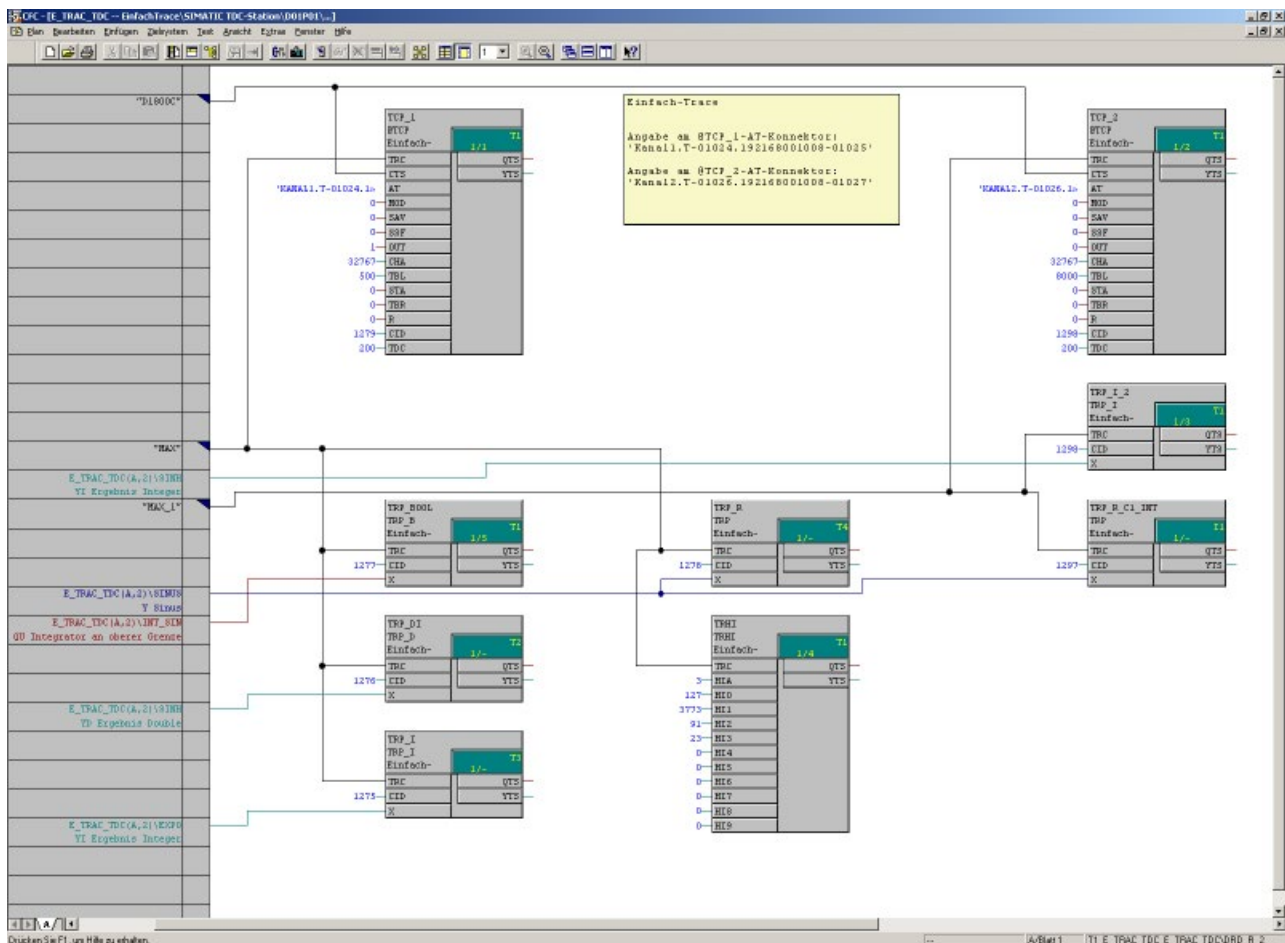


Figure 3-112 Example of the configuration of two Simple Traces

3.16.1.5 Response frames

The response frame generated by the @TCP FB depends on the value set at its OUT connector. With the connector value 0, single frames are generated based on the same output format as for system traces. In single frame mode, an acquisition block is selected for the output of trace values in one or several frames, based on the setting at connector CID of the @TCP FB. A group frame is generated if the value at connector OUT is set to 1. A group frame is characterized by the fact that it contains all trace values of all acquisition blocks. In this case, it is not necessary to define any settings at connector CID of the @TRP FB.

Single frames

An single frame is generated by the Simple Trace function if the value at initialization connector OUT of @TCP is set to 0. Resultant frame structure:

Byte no.	Content	Data format	Significance
1, 2	Connector ID code	unsigned int16	
3	Output format	char	0=binary, 1=standardized
4	Connector interpretation	char	1 - 12
5 to 10	Last acquisition time	6 chars	
11, 12	Packet no.	unsigned int16	
13, 14	Number of packets	unsigned int16	
15	Logging mode	char	0=cyclic, 1=interrupt task
16	Connector format	char	
17 to 20	Sampling time	unsigned int32	in 1/10 ms
21, 22	Task ID	unsigned int16	
23, 24	Number of blocks in trace buffer	unsigned int16	
25, 26	Number of blocks in frame	unsigned int16	
27, 28	empty	unsigned int16	

Note

Frame body: As of 29 trace value blocks. See the following description.

The **connector ID code** identifies the acquisition block that outputs its trace values in the frame. This specification is identical with that at the input connector CID of the @TCP FB.

Output format specifies the format of trace values in the trace value blocks. With binary format, the values of the connector values are always output according to their data type in standardized floating point format and with a length of 4 bytes.

The **connector interpretation** provides information on the type of recorded connector and may assume the following values:

Constant	Connector interpretation
1	N = standard Interpretation
3	I = integer
8	B = Boolean value

The last **acquisition time** relates to the last trace value logged, provided logging was executed in cyclic mode (this information is insignificant for non-cyclic logging). This trace value is the last trace value or trace value block in the frame body with package no. 0. The acquisition time is transferred in TDC format "Time and Date" (see "Compatible user data structures (Page 99)").

The frames are defined in successive order in the **Packet no.** and **Number of packets** fields. Each response frame returned for an (executed) output command is assigned two counters that contain the current packet number and the total number of packets that are necessary to transmit all logged values. The numbering convention specifies that the last packet is assigned packet no. 0. The following numbering scheme is derived, for example, from a scenario in which the trace values must be transmitted in three frames:

Example:

	1st Response frame	2nd Response frame	3rd Response frame
Packet no.	2	1	0
Number of packets	2	2	2

The **logging mode** is used to provide the information as to whether the corresponding acquisition block was configured in a cyclic task or interrupt task.

The **connector format** defines the size of the trace values logged and may assume the following values:

Constant	Significance
1	1 byte (logged by TRP_B block)
2	2nd bytes (logged by TRP_B_I block)
4	4 bytes (logged by TRP_D block)
5	4 bytes standardized (logged by TRP block)

The **sampling time** defines the sampling time of the corresponding acquisition block in 0.1 ms.

The **task ID** provides information on the task level at which the corresponding acquisition blocks was configured:

Sampling time	Constant
T1	0
T2	1
T3	2
T4	3
T5	4

Alarm level	Constant
I1	5
I2	6
I3	7
I4	8
I5	9

The **number of trace value blocks in the trace buffer** defines number of values written to the trace buffer for this acquisition block.

The **number of trace value blocks in the frame** defines the number of trace value blocks the current frame contains.

Trace value blocks contains the data logged for a connector at a specific time (possibly expanded with dummy bytes). The structure of a trace value block depends on the connector format, on the logging mode of the acquisition block, and on the configured output format of the @TCP FB. The following table lists all possible trace value blocks (all numbers in bytes).

Logging mode	Output format	Connector format	Connector value	Dummy bytes	Logging time	Dummy bytes	Block length
cyclic	0	1	1	-	-	-	1
cyclic	0	2	2	-	-	-	2
cyclic	0	4	4	-	-	-	4
cyclic	0	5	4	-	-	-	4
non-cyclic*	0	1	1	1	6	-	8
non-cyclic*	0	2	2	-	6	-	8
non-cyclic*	0	4	4	-	6	2	12
non-cyclic*	0	5	4	-	6	2	12
cyclic	0	1,2,4,5	4	-	-	-	4
non-cyclic*	0	1,2,4,5	4	-	6	2	12

* The connector value is logged by an acquisition block at interrupt level (I1--I5).

A trace value block consists of the connector value, a dummy byte (optional), the time of acquisition (only for non-cyclic logging, output in TDC "Time and Date" format, as well as additional dummy bytes (optional). The output format is determined by the setting at connector SSF of the @TCP block. The connector format is determined by the acquisition block type.

Example:

Structuring of the values in the frame body for a single-byte connector that was logged by an acquisition block on non-cyclic mode (interrupt level). The values are output in standardized floating point format.

Byte	Significance
29...32	Trace value at the time t1
33...38	Acquisition time t1
39, 40	Dummies
41...44	Trace value at the time t2
45...50	Acquisition time t2
51, 52	Dummies
...	...

Group frames

The @TCP FB generates a group frame if the value at initialization connector OUT of the @TCP FB is set to 1.

A group frame always consists of **only one** frame. As a consequence, the size of the output channel must be configured appropriately to enable transmission of all trace values in a single frame (CHA connector of the @TCP FB). The @TCP FB will enter the OFF state at the start of the cyclic phase if the configured channel size is insufficient. In this phase, the block generates a communication error message that includes a comment specifying the minimum channel size. The minimum size of the output channel can be calculated based on the following equation:

$$CHA \geq 12 + 4 \times HIA + 12 \times TRPx + 4 \times TBL$$

Meaning of the abbreviations:

- CHA - minimum channel size in bytes (set at connector CHA of the @TCP)
- HIA - specification at connector HIA of the TRHI FB (0 if the TRHI FB is not configured)
- TRPx - number of configured (and properly initialized) acquisition blocks
- TBL - size of the trace buffer (specified at connector TBL of the @TCP FB)

The group frame structure is as follows:

Frame header:

Byte no.	Content	Data format	Significance
1	Number of active TRPx	char	
2	Output format	char	See single frame
3 to 8	Last acquisition time	6 chars	
9 to 12	Parameter 0	unsigned int32	first parameter TRHI--FB
...			
9 + (n-1) x 4	Parameter n-1	unsigned int32	n-th parameter TRHI FB

Frame body:

Byte no.	Content	Data format	Significance
9 + n * 4	Connector ID code	unsigned int16	See single frame
11 + n * 4	Connector interpretation	char	See single frame
12 + n * 4	Connector format	char	See single frame
13 + n * 4	Sampling time	unsigned int32	See single frame
17 + n * 4	Task ID	unsigned int16	See single frame
19 + n * 4	Number of trace value blocks	unsigned int16	See single frame
21 + n * 4	Trace value blocks		See single frame

A frame body exists for each acquisition block. The frame bodies are sorted according to the registration sequence of the corresponding acquisition blocks.

The significance of data in group frames is basically identical to the type of data in single frames.

The **number of active TRPx** corresponds to the number of frame bodies in the group frame. The frame bodies can then be assigned to the acquisition blocks based on the connector ID code the corresponds to the CID definition of the acquisition blocks.

The **last acquisition time** is entered in the header of a group frame when the stop signal is detected. The last acquisition time in single frames corresponds to the time of the last trace value entry in the trace buffer and can therefore differ in acquisition blocks.

The **parameters 1 - n** of the TRHI FB are only entered if this block was actually configured. Connector HIA of the TRHI FB indicates the number of parameters to insert in the header.

Caution: **The group frame does not contain information of whether and how many parameters have been transferred from the TRHI FB.** Interpretation of the group frame therefore depends on the configuration. The parameters return additional data of which the scope and interpretation is can be freely selected by users.

Note

The group frame does not contain information of whether and how many parameters have been transferred from the TRHI FB. Interpretation of the group frame therefore depends on the configuration. The parameters return additional data of which the scope and interpretation is can be freely selected by users.

Each frame body starts with an offset which is divisible by 4 with regard to the frame start. For this reason, frame bodies may be padded with dummy bytes.

Example: A group frame containing three trace values of a TRP_B FB is generated. The TRP_B FB is operated in cyclic logging mode and binary output format, so that the corresponding trace value block has a size of one byte. The number of trace value blocks field in the message frame body then contains the value 3, is followed by three bytes of trace value blocks. A dummy byte is now inserted and followed with the start of the next frame body. No dummy bytes are inserted after the last frame body.

3.17 TDC-OPC server connection

3.17.1 Overview

Introduction

OPC is a vendor-independent software interface that allows data exchange between hardware and software from different vendors.

OPC components that supply data are called OPC servers. They implement the interfacing to existing communication systems. In addition to services, they provide information for the OPC client from any data source. Each OPC server has a unique name to identify it.

System data and events of the SIMATIC TDC multiprocessor control system can be monitored, accessed and processed from the PC via the OPC interface.

Note

Increased communication load

Depending on settings, PG/PC tools such as WinCC or CFC online and HMI Panels create an increased communication load.

This can result in longer response times and communication failures.

Additional information

- SIMATIC NET Industrial Communication with PG/PC Volume 1 Basics
(<http://support.automation.siemens.com/WW/view/en/77376110>)
- SIMATIC NET Industrial Communication with PG/PC Volume 2 Interfaces
(<http://support.automation.siemens.com/WW/view/en/77378184>)
- SIMATIC NET Commissioning PC Stations
(<http://support.automation.siemens.com/WW/view/en/109488960>)

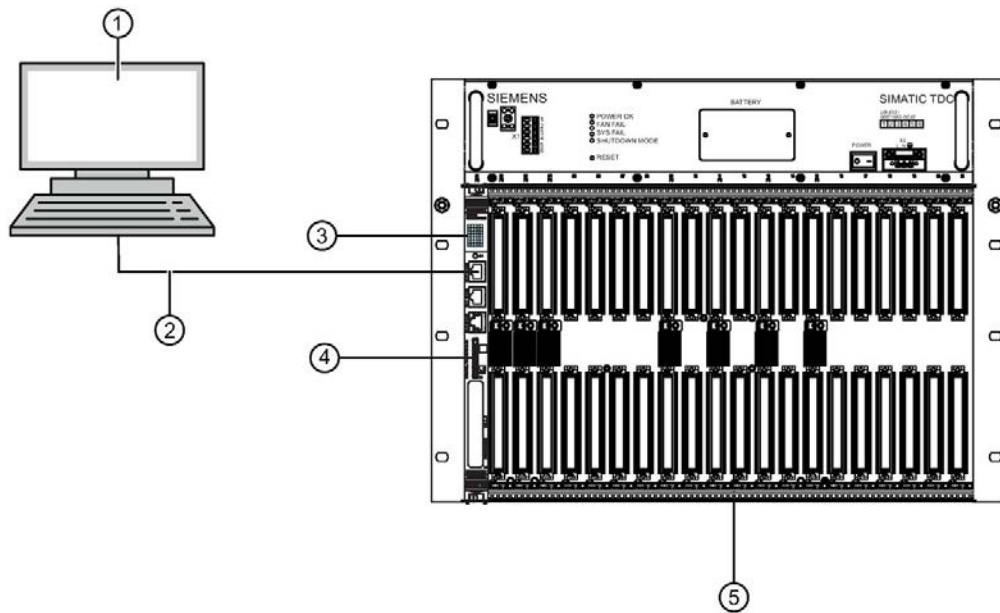
3.17.2 Configuration model

The structure of these configuration instructions reflects the chronological order of steps to take for creating the entire configuration.

Appropriate knowledge of SIMATIC Manager and about the configuration of SIMATIC TDC is presumed.

Hardware

The following devices or components were selected and for the configuration model and positioned as follows :



- ① PC station
- ② Ethernet cable
- ③ CPU module CPU555
- ④ Program memory card MMC
- ⑤ Rack UR6021

Figure 3-113 Configuration model

3.17.3 Configuring an OPC server connection

Procedure

1. Add a "SIMATIC PC station" to your project.
2. Add an OPC server to the SIMATIC PC station.
3. Add an Industrial Ethernet "IE General" module to the SIMATIC PC station.

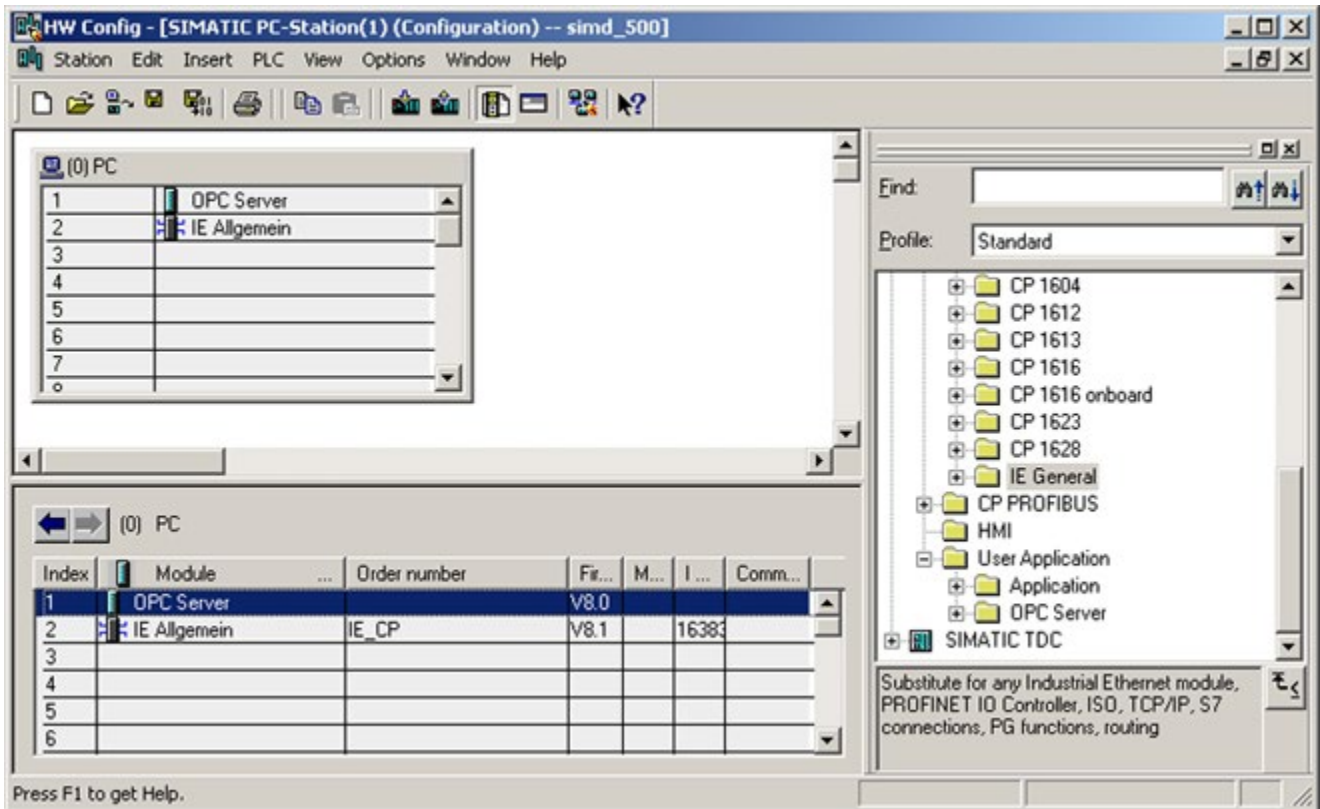


Figure 3-114 Complete the Industrial Ethernet module

4. Configure the OPC server in the Station Configuration Editor.
5. Configure the Industrial Ethernet module in the Station Configuration Editor.

6. Insert an unspecified S7 connection in NetPro.

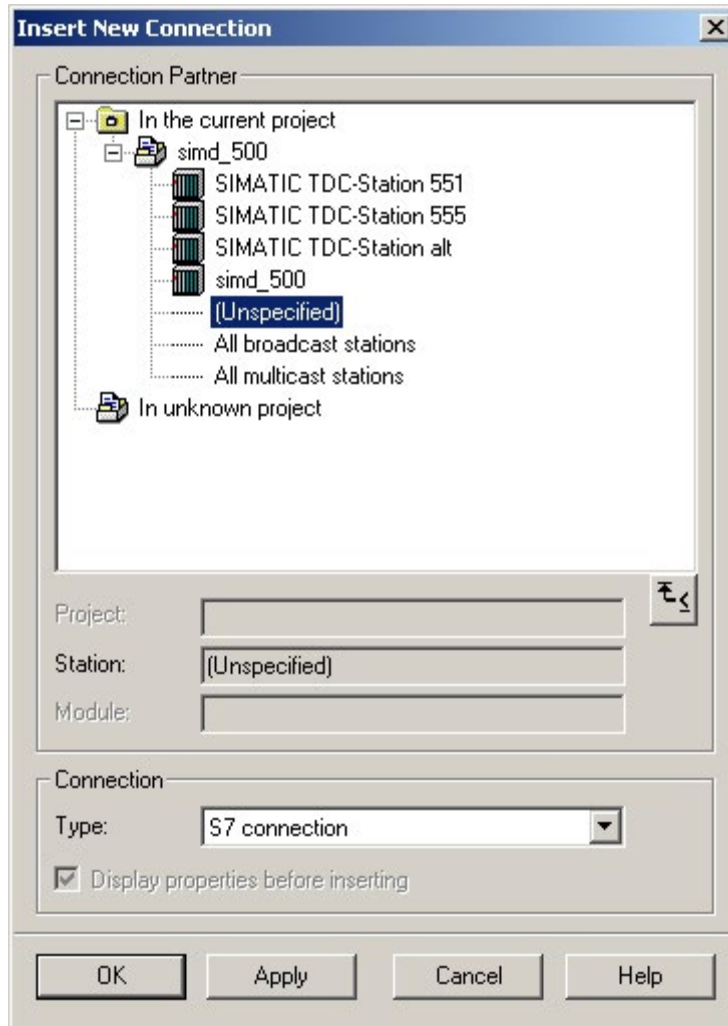


Figure 3-115 Insert unspecified S7 connection

7. Configure this connection in the properties dialog of the S7 connection ("General" tab).

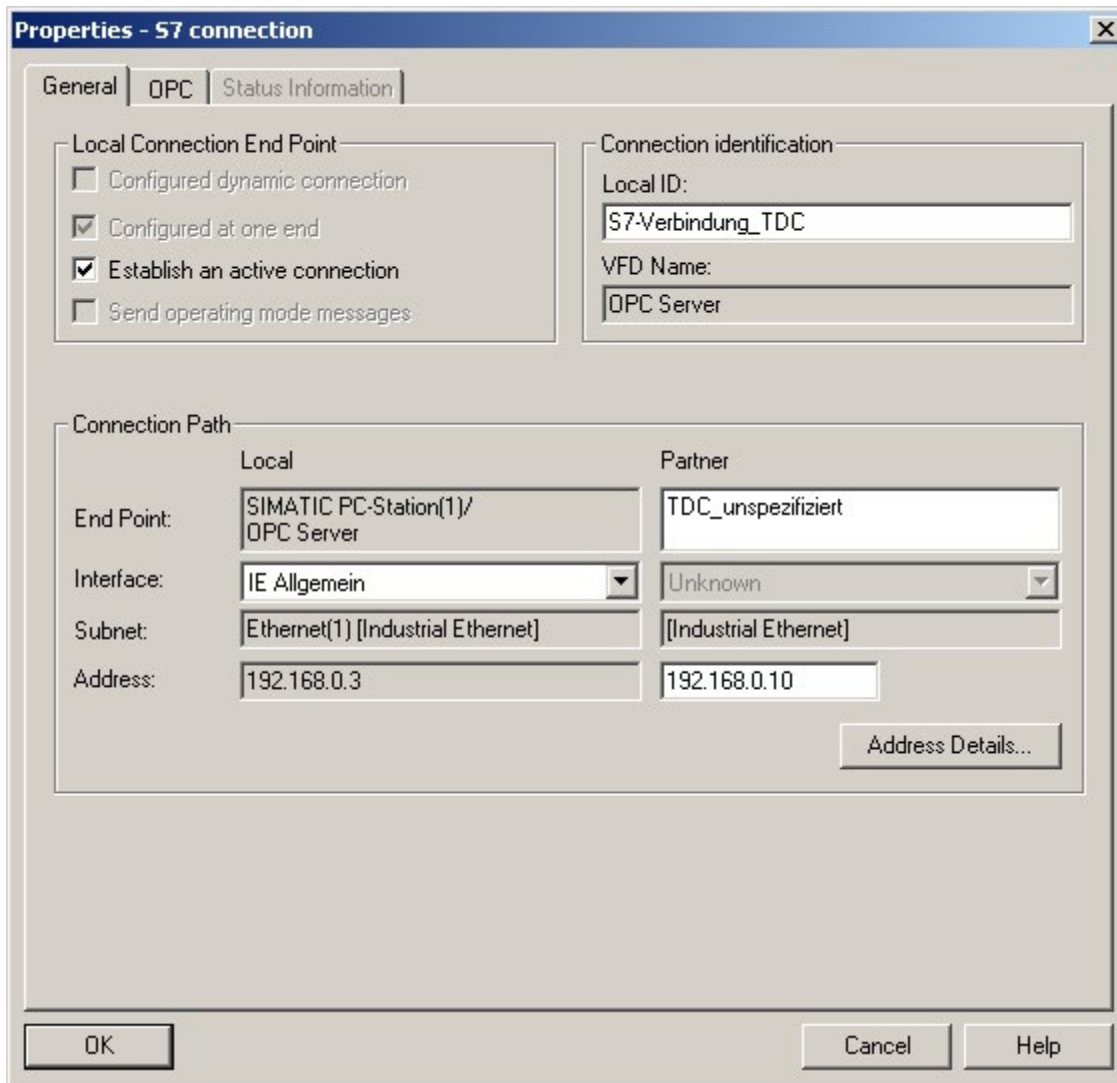


Figure 3-116 Configure the properties of the S7 connection

8. If necessary, adapt the OPC properties ("OPC" tab).
9. Download the PC station in the Station Configuration Editor.

10. Test the connection:

- Open the OPC Scout.
- Add a new DA view.
- Create a new item that has the following structure in keeping with the previous example: "S7:[S7-connection_TDC]DB1,INT1"
The DB1.DBW0 of the type Integer is thus accessed via the previously created, unspecified S7 connection "S7-connection_TD".

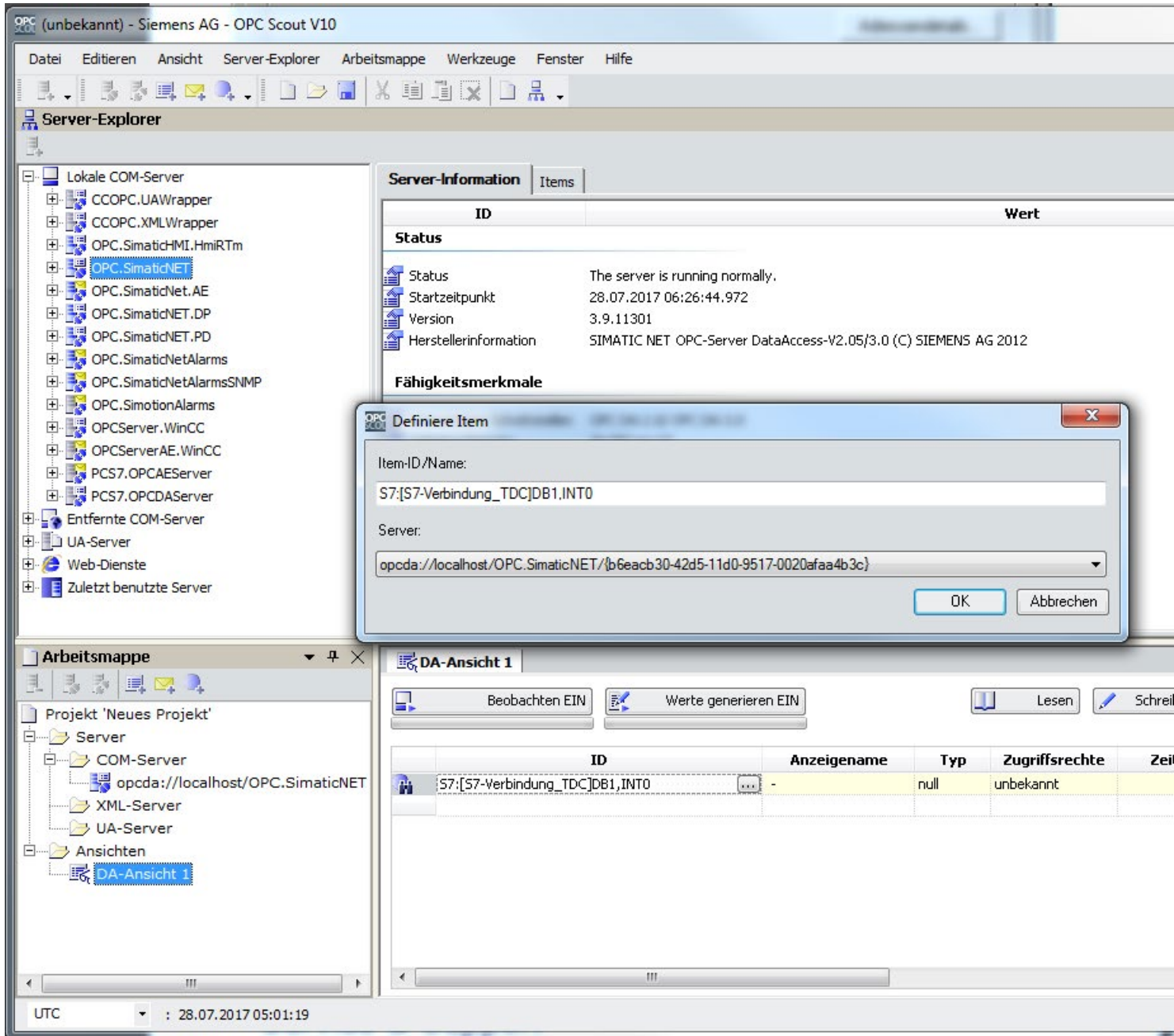


Figure 3-117 Create item

- Change to monitoring mode.
Refer to the OPC Scout help for the exact structure of the item ID. You can also find information on operating OPC Scout in this help documentation.

Note

Series 551 modules

The S7CON block needs to be configured if you wish to access a series 551 module (OBA1, OBA2, OBA3) with OPC.

Function blocks example in D7-SYS

In D7-SYS, an S7 DB with DB1.DBW0 as integer value must be configured on the TDC CPU555.

The following figure shows an example of two interconnected function blocks.

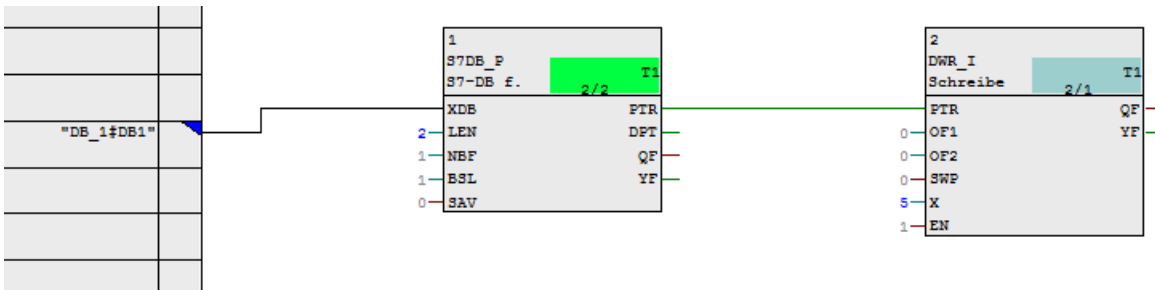


Figure 3-118 Function blocks example

You can also use the DBs mentioned in section WinCC connection to SIMATIC TDC via standard channel (SIMATIC S7 Protocol Suite.CHN) (Page 239) for WinCC communication from the address book.

3.18 Connections across network boundaries (routing)

Possibilities for use

For SIMATIC D7-SYS V8.2 with the appropriate hardware, routing is possible across racks. If you are using routing-capable partners, further routing from these partners is also possible. In addition, routing to S7 hardware that is behind a rack or between two racks is possible.

CPU555 V1.1 and CP51M1, with up-to-date firmware in each case, can be used for Profinet routing. CP50M1 with up-to-date firmware can be used for PROFIBUS routing.

You have the following options for using routing for TDC modules:

- Routed download to CPU555 V1.1
- Routed reading of module states
- Routed PROFINET IO diagnostics
- Routed OCM in CFC/SFC
- Routed OCM with HMI and WCF2008
- Use of the "PC internal.local.1" PG/PC interface for the PC station (with STEP 7 V5.5 SP4 HF14 or STEP 7 V5.6 HF3)
- OPC access to S7 CPUs routed via TDC hardware

These options also apply if you configure S7 hardware (for example S7-300) in NetPro.

The following functions are not supported at present:

- Routed OPC communication with CPU555 V1.1 modules
- Routed PROFIBUS DP diagnostics
- Creation of specified connections in NetPro

Note

When using routing functionality, be aware of the communication load, because this can lead to connection failures.

The required configuration for the routing is performed in NetPro and, if necessary, also in the topology editor. You can find more information on S7 routing in the Programming with STEP 7 manual on the Internet

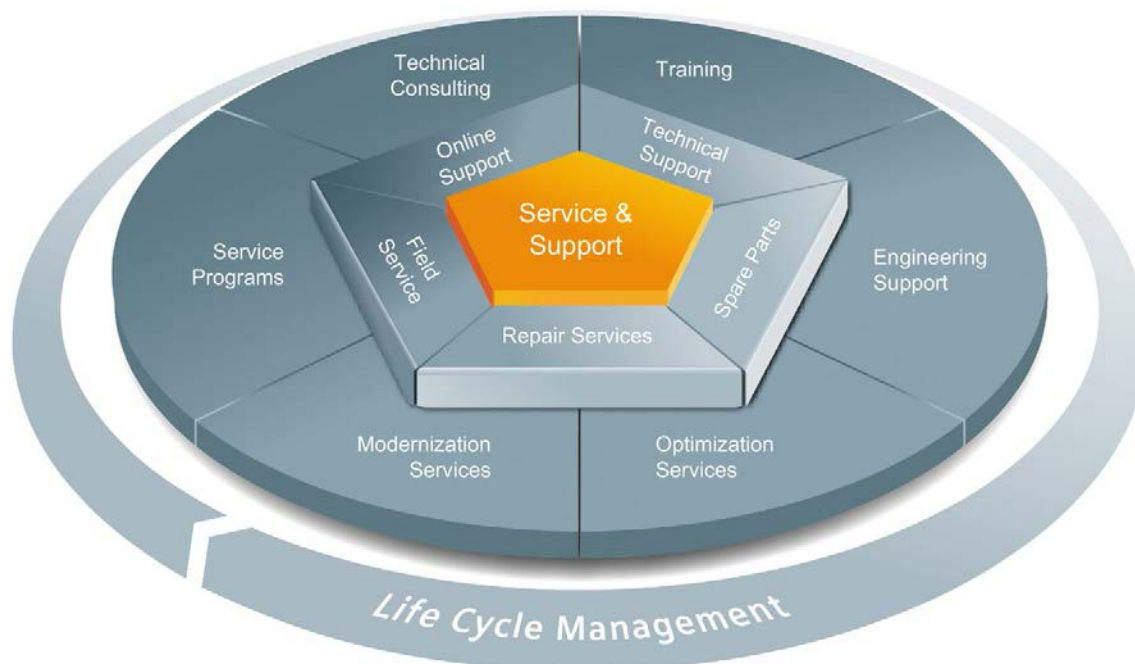
(<https://support.industry.siemens.com/cs/ww/en/view/109751825>).

Note

When using the routing functionality, you need to define "Diagnostics header scan" as the search method for topology detection for the topology editor in online mode (Topology editor > Options).

Service & Support

A.1 Service & Support



The unmatched complete service for the entire life cycle

For machine constructors, solution providers and plant operators: The service offering from Siemens Industry Automation and Drive Technologies includes comprehensive services for a wide range of different users in all sectors of the manufacturing and process industry.

To accompany our products and systems, we offer integrated and structured services that provide valuable support in every phase of the life cycle of your machine or plant – from planning and implementation through commissioning as far as maintenance and modernization.

Our Service & Support accompanies you worldwide in all matters concerning automation and drive technology from Siemens. We provide direct on-site support in more than 100 countries through all phases of the life cycle of your machines and plants.

You have an experienced team of specialists at your side to provide active support and bundled know-how. Regular training courses and intensive contact among our employees – even across continents – ensure reliable service in the most diverse areas

Online Support

The comprehensive online information platform supports you in all aspects of our Service & Support at any time and from any location in the world.

You can find Online Support on the Internet at the following address: Internet (<http://www.siemens.com/automation/service&support>).

Technical Consulting

Support in planning and designing your project: From detailed actual-state analysis, definition of the goal and consultation on product and system questions right through to the creation of the automation solution.

Technical Support

Expert advice on technical questions with a wide range of demand-optimized services for all our products and systems.

You can find Technical Support on the Internet at the following address: Internet (<http://www.siemens.com/automation/support-request>).

Training

Extend your competitive edge – through practical know-how directly from the manufacturer.

You can find the training courses we offer on the Internet at the following address: Internet (<http://www.siemens.com/sitrain>).

Engineering Support

Support during project engineering and development with services fine-tuned to your requirements, from configuration through to implementation of an automation project.

Field Service

Our Field Service offers you services for commissioning and maintenance – to ensure that your machines and plants are always available.

Spare parts

In every sector worldwide, plants and systems are required to operate with constantly increasing reliability. We will provide you with the support you need to prevent a standstill from occurring in the first place: with a worldwide network and optimum logistics chains.

Repairs

Downtimes cause problems in the plant as well as unnecessary costs. We can help you to reduce both to a minimum – with our worldwide repair facilities.

Optimization

During the service life of machines and plants, there is often a great potential for increasing productivity or reducing costs.

To help you achieve this potential, we are offering a complete range of optimization services.

Modernization

You can also rely on our support when it comes to modernization – with comprehensive services from the planning phase all the way to commissioning.

Service programs

Our service programs are selected service packages for an automation and drives system or product group. The individual services are coordinated with each other to ensure smooth coverage of the entire life cycle and support optimum use of your products and systems.

The services of a Service Program can be flexibly adapted at any time and used separately.

Examples of service programs:

- Service contracts
- Plant IT Security Services
- Life Cycle Services for Drive Engineering
- SIMATIC PCS 7 Life Cycle Services
- SINUMERIK Manufacturing Excellence
- SIMATIC Remote Support Services

Advantages at a glance:

- Reduced downtimes for increased productivity
- Optimized maintenance costs due to a tailored scope of services
- Costs that can be calculated and therefore planned
- Service reliability due to guaranteed response times and spare part delivery times
- Customer service personnel will be supported and relieved of additional tasks
- Comprehensive service from a single source, fewer interfaces and greater expertise

Contact

At your service locally, around the globe: your partner for consultation, sales, training, service, support, spare parts... for the entire range of products supplied by Industry Automation and Drive Technologies.

You can find your personal contact in our contacts database at: Internet (<http://www.siemens.com/automation/partner>).

Index

\$

\$ signals, 46

A

Alarm logging, 194

Alarm formats, 200, 204

Alarm type definition, 200

Alarms, 196, 199

Communications error message, 202

Error or alarm message, 199

Output format, 209

Overflow alarm, 202

System alarm, 203

System error, 203

Alarm loggingAlarm logging

LoggingLogikAlarmLoggingBlocks, 195

Alarm-driven processing, 67

B

Basic clock cycle, 55

Basic CPU clock cycle/Task Manager, 48

Behavior in the event of a fault, 71

Buffer memory coupling, 105

C

CFC chart (Continuous Function Chart), 34

CFC editor, 34

Margins, 39

Parameterization dialogs, 35

Communication

WinCC via SIMATIC NET, 278

Communication basics, 79

Communication blocks

Address I/Os AT, AR, US, 87

Central coupling blocks, 96

Firmware status, ECL, ECO connection, 94

I/O MOD, 89

Initialization I/O CTS, 85

Status display, output YTS, 94

Transmitters and receivers, 97

Communication utilities

Overview, 80

Consistency check, 33

Coupling modules

Number of modules in the rack, 101

Couplings

Data interface, 101

Functional principle, 95

Overview, 81

User data structures, 99

CP52M0 rack coupling

Areas of application, 107

Configuration, 109

Performance data, 113

Power on/off response, 108

CP53M0 rack coupling

Acknowledgment, 119

Configuration, 121

General, 114

Hardware installation, 117

Initialization and monitoring, 115

Restart capability, 121

Restrictions, 122

Scope of performance, 117

Shutdown of a coupling partner, 118

Switching on the master rack, 119

CPU synchronization, 54

Configuring the CPU basic clock cycle, 56

Configuring the interrupt task, 57

Hardware timer response to failure, 55

Time synchronization, 54

Cycle errors, 60

Eliminating, 62

D

Data consistency, 44, 50

Deadtimes, 47

Diagnostics function block, 142

Direct CPU-CPU coupling, 103

Download in RUN, 43

E

Error diagnostics

Background processing, 73

Error differentiation, 68

- F**
 - Fast \$ signal, 47
 - Function block, 34
 - Assigning blocks to cyclic tasks, 36
 - Assignment to interrupt tasks, 36
 - Parameterizing and interconnecting, 35
 - Pseudo comments / comments, 39
 - Selecting, 34

- H**
 - Hardware address, 39
 - Hardware modules, 31
 - Parameterizing, 32
 - Selecting, 31
 - HW Config, 32
 - Parameterization dialogs, 32

- I**
 - Initialization, 65
 - Interconnecting, 39
 - Interrupt task, 62
 - Isochronous mode and constant bus cycle time, 137

- L**
 - Libraries, 30
 - Limited number of interconnections, 49
 - Loading the user program, 40
 - Offline download, 40
 - Online download, 40
 - Local couplings in the rack, 102
 - Local CPU coupling, 102

- M**
 - MPI coupling, 145
 - Configuration, 145

- N**
 - Name assignment, 30

- O**
 - Operating system components, 65
 - Overflow, 62

- P**
 - Performance features, 63
 - Alarm tasks, 64
 - Computing times of the operating system, 64
 - Cyclic tasks, 63
 - PNIO communication, 146
 - Pointer-based communication blocks
 - Applications, 223
 - Corresponding function blocks, 225
 - Examples, 227
 - Features, 224
 - Function principles, 222
 - Introduction, 222
 - Notes on configuration, 226
 - Pointer interface, 225
 - Process data, 211
 - Blocks CRV, CTV, 215
 - Channel marshalling blocks, 216
 - Channels, 221
 - Diagnostics, 219
 - Distribution block, 218
 - Function blocks, 211
 - Sample configuration, 213
 - Virtual connections, 211
 - Process image, 50
 - Cyclic tasks, 52
 - Implementation, 51
 - Interrupt tasks, 53
 - PROFIBUS DP
 - Address connection, 135
 - Configuration, 134
 - Error class, 144
 - SIEMENS DP slaves, 143
 - SYNC/FREEZE, 137
 - PROFIBUS DP coupling
 - Central coupling block, 134
 - Communication service, 134
 - Data specified at address connection AT, AR, 135
 - Function blocks, 134
 - Properties, 133
 - Transfer mode, 134

- S**
 - Service, 232
 - SER function block, 234
 - System load, 235
 - Service utility
 - SER function block, 70
 - System load, response times, 71

- Seven-segment display, 76
 - Acknowledge error, 76
- Signal transfer, 44
 - Task, 44
- SIMATIC TDC
 - Configuring the station, 31
- Slot number, 32
- Standard mode, 52, 52
- SYNC/FREEZE configuration variants, 138
- System chart, 33
- System mode, 51

T

- Table function, 156
- Task Manager, 60
- Task processing, 48, 50
- Transmission mode
 - Handshake, 89
 - Image, 93
 - Multiple, 92
 - Overview, 89
 - Refresh, 90
 - Select, 91

U

- User stop system state, 43
- Utility programs, 69

W

- WinCC integration, 239
 - Configuration using the D7-SYS-OS Engineering Tool, 268
 - Coupling over TCP/IP with OCM functions, 241
 - MPI and PROFIBUS DP coupling variants, 259
 - S7DB configuration variant, 258

