SIEMENS

SIMATIC TDC

System- and communication configuring D7-SYS

Manual

Edition 07/2009
A5E01115023-04

Preface, Contents
The first project in a few steps 1
Systemsoftware 2
Communications configuring 3
Index

Safety guidelines

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 to property damage only have no safety alert symbol. The 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
with a safety alert symbol indicates that minor personal injury can result if proper precautions are not taken.

Caution
without a safety alert symbol indicates that property damage can result if proper precautions are not taken.

Attention
indicates that an unintended result or situation can occur if the corresponding notice is not taken into account.

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 awarning relating to property damage.

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by **qualified personnel.** Within the context of the safety notices in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Prescribed Usage

Note the following:

Warning
This device and its components may only be used for the applications described in the catalog or the technical description, and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens. Correct, reliable operation of the product requires proper transport, storage, positioning and assembly as well as careful operation and maintenance.

Trademarks

All names identified by ® are registered trademarks of the 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.

이 기기는 업무용(A급) 전자파 적합기기로서 판매자 또는 사용자는 이 점을 주의하시기 바라며 가정 외의 지역에서 사용하는 것을 목적으로 합니다.

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.

Siemens AG
Automation and Drives
Geschäftsgebiet Industrial Automation Systems
Postfach 4848, D- 90327 Nürnberg

Siemens Aktiengesellschaft

A5E01115023-04

Preface

Purpose of this Manual

This Manual explains the principle use and functions of the D7-SYS automation software with the main focus on the appropriate technological and drive control components SIMATIC TDC, FM 458-1 DP, T400, SIMADYN D.

TDC: Technology and Drives Control

Basic knowledge required

This Manual addresses programmers and commissioning engineers. General knowhow regarding automation technology is required in order to understand the contents of the Manual.

Validity of the Manual

This Manual is valid for SIMATIC D7-SYS Version 7.1 SP1.

Information overview

This manual is part of the overall documentation for the technological and drive control components T400, FM 458, SIMADYN D, SIMATIC TDC and SIMATIC D7-SYS.

Titel	Inhalt
System and communications configuring D7-SYS	**The first project in a few steps** This Section provides an extremely simple entry into the methodology when assembling and programming the SIMATIC TDC/SIMADYN D control system. It is especially conceived for first-time users of a control system. **System software** This Section provides basic know-how about the structure of the operating system and an application program of a CPU. It should be used to obtain an overview of the programming methodology, and basis for configuring user programs. **Communications configuring** This section provides you with basic know-how about the communication possibilities and how you configure links to the communication partners. **Changeover from STRUC V4.x to D7-SYS** Essential features are included in this section, which have changed over STRUC V4.x with the introduction of SIMATIC D7-SYS.
D7-SYS - STEP 7, CFC and SFC configuring	**Basis software** This section explains the essential use and the functions of the STEP 7 automation software. For first users, it provides an overview on configuring, programming and commissioning a station. When working with the basis software, you can access the online help which provides you with support when it comes to detailed questions on using the software. **CFC** The CFC language (Continuous Function Chart) allows you to graphically interconnect blocks. When working with the particular software, you can also use the online help which can answer detailed questions regarding the use of the editors/compiler. **SFC** Configuring sequence controls using SFC (Sequential Function Chart) of SIMATIC S7. In the SFC editor, you generate a sequence chart using graphic resources. The SFC elements of the chart are then positioned according to specific rules.
Hardware	The complete hardware spectrum is described as reference in this Manuals.

Titel	Inhalt
Selecting Function blocks	The Reference Manual provides you with an overview of all of the function blocks for the corresponding technology and drive control components - SIMATIC TDC, FM 458-1 DP, SIMADYN D and T400. **Chapter 1** The function blocks that can be configured in **all target systems** of SIMATIC D7-SYS are described in this Chapter. **Chapter 2** The function blocks that can **only** be configured for SIMATIC TDC are described in this Chapter. **Chapter 3** The function blocks that can only **only** be configured for FM 458-1 DP application module are described in this Chapter. **Chapter 4** The function blocks that can **only** be configured for SIMADYN D and T400 are described in this Chapter.

Guide

As first time user, we recommend that this Manual is used as follows:

- Please read the first section on using the software in order to get to know some of the terminology and basic procedure.
- Then use the particular sections of the Manual if you wish to carry-out certain processing steps (e.g. loading programs).

If you have already executed a small project, and have gained some experience, then you can read individual sections of the Manual in order to get up to speed about a specific subject.

Special Notes

This user part of the Manual does not include any detailed information/instructions with individual descriptions, but is only intended to provide a basic procedure. More detailed information on the dialog boxes in the software and how they are handled is provided in the appropriate online help.

Training Center

We offer courses to help you get started with the S7 automation system. Contact your regional training center or the central training center in D 90327 Nuremberg, Federal Republic of Germany.

Internet: http://www.sitrain.com

Service & Support im Internet

In addition to our paper documentation, our complete knowledge base is available to you on the Internet at:

http://www.siemens.com/automation/service&support

There, you will find the following information:

- Newsletters providing the latest information on your products
- A search engine in Service & Support for locating the documents you need
- A forum where users and experts from all over the world exchange ideas
- Your local contact partner for Automation & Drives in our Contact Partners database
- Information about on-site service, repairs, spare parts, and much more under "Services"

Contents

Preface ... **iii**

1 In just a few steps to the first project ... **1-1**

1.1 Prerequisites ... 1-2
1.1.1 Software and hardware ... 1-2
1.1.2 What you can expect ... 1-4
1.2 Creating a new project ... 1-5
1.3 Defining the hardware ... 1-5
1.4 Generating a CFC chart ... 1-6
1.4.1 Generating a new chart ... 1-6
1.4.2 Inserting, parameterizing and inter-connecting function blocks ... 1-6
1.5 Testing, compiling and downloading the project ... 1-10
1.5.1 Checking the project consistency and compiling ... 1-10
1.5.2 Downloading the user project into the SIMATIC TDC-CPU module ... 1-10
1.6 Testing the user project ... 1-12
1.6.1 Disconnecting the connection online ... 1-13
1.6.2 Generating a connection online ... 1-13
1.6.3 Changing the parameterization online ... 1-13
1.6.4 Inserting a block online ... 1-13
1.6.5 Deleting blocks online ... 1-13
1.7 Results ... 1-14
1.8 Archiving the project ... 1-14

2 Systemsoftware ... **2-1**

2.1 Configuring ... 2-2
2.1.1 General description ... 2-2
2.1.1.1 Configuring tools ... 2-2
2.1.1.2 Configuring steps ... 2-3
2.1.1.3 Terminology and libraries ... 2-3
2.1.2 Configuring the hardware ... 2-4
2.1.2.1 The first step: Selecting the hardware modules ... 2-4
2.1.2.2 The second step: Parameterizing the hardware modules ... 2-5
2.1.2.3 The third step: Checking the configuring ... 2-7
2.1.3 Creating CFC charts ... 2-7

2.1.3.1 The first step: Selecting the function blocks 2-7
2.1.3.2 The second step: Parameterizing and interconnecting function blocks 2-8
2.1.3.3 The third step: Compiling and loading the user program into the CPU 2-13
2.1.4 Operating statuses of a CPU module 2-14
2.1.5 Description and use of the signal transfer mechanisms 2-15
2.1.5.1 Data consistency 2-15
2.1.5.2 Data transfer within the same task of a CPU 2-16
2.1.5.3 Data transfer between various CPU tasks 2-16
2.1.5.4 Data transfer between cyclic tasks of several CPUs 2-17
2.1.5.5 Data transfer between interrupt tasks of several CPUs 2-18
2.1.5.6 Minimizing the deadtimes 2-19
2.1.5.7 Processing sequence within a basic CPU clock cycle 2-19
2.1.5.8 Interconnection changes and limited number of interconnections 2-20
2.1.6 Significance and uses of the process image 2-21
2.1.6.1 Implementing the process image 2-22
2.1.6.2 Process image for cyclic tasks 2-23
2.1.6.3 Process image for interrupt tasks 2-24
2.1.7 Significance and application of the CPU synchronization 2-25
2.1.7.1 Time synchronization 2-25
2.1.7.2 Synchronizing its own basic clock cycle to the basic clock cycle of a master CPU 2-25
2.1.7.3 Synchronizing its own basic clock cycle to an interrupt task of a master CPU 2-26
2.1.7.4 Synchronizing its own interrupt tasks to interrupt tasks of a master CPU 2-26
2.1.7.5 Synchronizing several SIMATIC TDC/SIMADYN D stations 2-26
2.1.7.6 Response when the synchronization fails 2-26
2.1.7.7 Configuring the CPU basic clock cycle synchronization 2-26
2.1.7.8 Configuring the interrupt task synchronization 2-28
2.1.7.9 Example of a synchronization configuration 2-29
2.1.8 Significance of the processor utilization 2-30
2.1.8.1 Determining the approximate processor utilization 2-30
2.1.8.2 Calculating the precise processor utilization 2-30
2.1.8.3 Mode of operation of the task administrator 2-31
2.1.8.4 Eliminating cycle errors 2-32
2.1.9 Technical data of the operating system 2-33
2.1.9.1 Features 2-33
2.1.9.2 The basic operating system functions 2-34
2.1.9.3 The service utility 2-37
2.2 Function description and user instructions 2-40
2.2.1 Fatal system error "H" 2-40
2.2.2 Background processing 2-43
2.2.2.1 Online test mode 2-43
2.3 System chart @SIMD 2-44

3 Communications configuring 3-1

3.1 Introduction 3-2
3.1.1 Basic information on communications 3-2
3.1.1.1 Overview of the various data couplings 3-2
3.1.2 Overview of the communication utilities 3-6
3.1.3 Communication block I/Os 3-7
3.1.3.1 Initialization input CTS 3-7

3.1.3.2 Address connections AT, AR and US ... 3-7
3.1.3.3 Data transfer mode, MOD input ... 3-8
3.1.3.4 Firmware status, ECL, ECO connection ... 3-12
3.1.3.5 Status display, output YTS ... 3-13
3.1.4 Mode of operation of the couplings ... 3-13
3.1.4.1 Central coupling blocks ... 3-14
3.1.4.2 Transmitters and receivers ... 3-15
3.1.4.3 Compatible net data structures ... 3-16
3.1.4.4 Number of coupling modules in a subrack ... 3-17
3.1.4.5 Reorganizing a data interface ... 3-18
3.2 Couplings on the subrack ... 3-19
3.2.1 Local CPU coupling ... 3-19
3.2.2 Direct CPU-CPU coupling ... 3-20
3.2.3 Communications buffer coupling ... 3-21
3.3 Subrack coupling CP52M0 ... 3-22
3.3.1 Applications ... 3-22
3.3.2 Behavior when powering-up and powering-down ... 3-23
3.3.3 Synchronizing and triggering methods ... 3-24
3.3.4 Configuring ... 3-24
3.3.4.1 Configuring in HWConfig ... 3-24
3.3.4.2 Configuring in CFC ... 3-26
3.3.5 Performance data ... 3-27
3.3.5.1 Data transfer rates ... 3-27
3.3.5.2 Cable lengths ... 3-27
3.3.5.3 Interface assignment ... 3-27
3.4 Subrack coupling CP53M0 ... 3-28
3.4.1 Hardware structure ... 3-31
3.4.2 Scope of supply ... 3-31
3.4.3 Response when "shutting down" a coupling partner ... 3-32
3.4.4 Response when "powering-up" the master subrack ... 3-32
3.4.4.1 Acknowledging ... 3-32
3.4.5 Restart capability ... 3-33
3.4.6 Configuring ... 3-34
3.4.7 Restrictions ... 3-35
3.5 TCP/IP coupling (CP51M1) ... 3-36
3.5.1 Comparison between TCP/IP and UDP ... 3-37
3.5.2 Typical configuration ... 3-38
3.5.3 Configuring steps ... 3-38
3.5.3.1 Configuring in HWConfig ... 3-38
3.5.3.2 Configuring with CFC ... 3-39
3.5.3.2.1 Central block @TCPIP ... 3-39
3.5.3.2.2 Receive block CRV ... 3-39
3.5.3.2.3 Send block CTV ... 3-40
3.5.4 Application information ... 3-42
3.5.4.1 Channel number ... 3-42
3.5.4.2 Telegram length ... 3-42
3.5.4.3 "Ping" on CP51M1 ... 3-42
3.5.4.4 Performance ... 3-42

3.5.5 Communications with WinCC ... 3-43
3.5.5.1 Standard coupling ... 3-43
3.5.5.2 Coupling via TDC PMC channel-DLL ... 3-43
3.5.6 Central service ... 3-43
3.5.7 Clock time synchronization ... 3-43
3.5.8 Changing-over from CP5100 to CP51M1 ... 3-43
3.6 TCP/IP coupling (CP5100) ... 3-45
3.6.1 Comparison between TCP/IP and UDP ... 3-46
3.6.2 Typical configuration ... 3-47
3.6.3 Configuring steps ... 3-48
3.6.3.1 Configuring in HWConfig ... 3-48
3.6.3.2 Configuring with CFC ... 3-49
3.6.3.2.1 Central block @TCPIP ... 3-49
3.6.3.2.2 Receive block CRV ... 3-49
3.6.3.2.3 Send block CTV ... 3-50
3.6.4 Application information ... 3-51
3.6.4.1 Channel number ... 3-51
3.6.4.2 Telegram length ... 3-52
3.6.4.3 "Ping" on CP5100 ... 3-52
3.6.4.4 Performance ... 3-52
3.7 PROFIBUS DP coupling (CP50M1) ... 3-53
3.7.1 General basics ... 3-53
3.7.2 Configuring ... 3-54
3.7.2.1 Configuring the DP system on CP50M1 ... 3-54
3.7.2.2 Configuring communications in CFC ... 3-54
3.7.2.3 Configuring as Slave ... 3-56
3.7.2.4 Shared Input ... 3-56
3.7.3 Equidistance ... 3-56
3.7.4 SYNC/FREEZE commands ... 3-56
3.7.4.1 Configuring versions of SYNC/FREEZE ... 3-57
3.7.5 Commissioning/ diagnostics ... 3-61
3.7.5.1 Diagnostics function block ... 3-61
3.7.5.2 Error class (ECL) and error code (ECO) ... 3-64
3.8 PROFIBUS DP coupling (CP50M0) ... 3-65
3.8.1 Configuring with D7-SYS ... 3-66
3.8.1.1 Central coupling block ... 3-66
3.8.1.2 Address connections AT, AR ... 3-67
3.8.1.3 SYNC/FREEZE commands ... 3-68
3.8.1.4 Configuring versions of SYNC/FREEZE ... 3-69
3.8.1.5 Diagnostics function block ... 3-73
3.8.2 Configuring with COM PROFIBUS ... 3-77
3.8.2.1 Harmonizing with data configured in CFC ... 3-77
3.8.2.2 CP50M0 as PROFIBUS slave ... 3-78
3.8.2.3 Loading the database ... 3-79
3.8.3 Start-up/diagnostics ... 3-80
3.8.3.1 LEDs ... 3-80
3.8.3.2 Error class (ECL) and error code (ECO) ... 3-81
3.8.3.3 Application example, PROFIBUS DP coupling ... 3-82
3.8.3.4 Typical configuration and system requirements ... 3-83

3.8.3.5 Check list of the required hardware and software components for SIMATIC TDC ... 3-84
3.8.3.6 Configuring under STEP 7 CFC ... 3-85
3.8.3.7 Using transmit- and receive blocks ... 3-87
3.8.3.8 Configuring the typical configuration in CFC ... 3-88
3.8.3.9 Configuring the SS52 communications module with COM PROFIBUS ... 3-91
3.8.3.10 Generating the COM database with COM PROFIBUS ... 3-91
3.8.3.11 Downloading the COM database into the CP50M0 ... 3-99
3.8.3.12 Working with the "SS52load" download tool ... 3-99
3.8.3.13 Behavior of the CP50M0 during and after the download ... 3-99
3.9 MPI coupling ... 3-101
3.9.1 Characteristics and hardware ... 3-101
3.9.2 Configuring ... 3-101
3.10 Table function ... 3-102
3.10.1 Introduction ... 3-102
3.10.1.1 Overview, "Manual mode" ... 3-103
3.10.1.2 Overview, "Automatic mode: Communications" ... 3-103
3.10.1.3 Function block WR_TAB ... 3-105
3.10.2 Manual mode ... 3-107
3.10.2.1 Application ... 3-107
3.10.2.2 Configuring ... 3-108
3.10.3 Automatic mode: Communications ... 3-109
3.10.3.1 Application with an S7 control and SIMATIC FM 458 application module ... 3-109
3.10.3.2 Configuring for S7 control and SIMATIC FM 458 application module ... 3-111
3.10.3.3 Inserting tabular values in the data block ... 3-112
3.10.3.3.1 Manually entering tabular values ... 3-112
3.10.3.3.2 Importing tabular values ... 3-116
3.10.3.3.3 Subsequently downloading tabular values into a DB ... 3-126
3.10.3.4 Structure of the data telegram for TCP/IP or DUST1 connection ... 3-127
3.10.4 Automatic mode: Memory card ... 3-128
3.10.4.1 Generating a table file in the csv format ... 3-129
3.10.4.2 Working with the D7-SYS additionalComponentBuilder ... 3-131
3.10.4.3 Downloading ... 3-134
3.10.4.4 Configuring the function blocks ... 3-136
3.11 Communications utility, message system ... 3-138
3.11.1 Entry logic of the message entry blocks ... 3-138
3.11.1.1 Message entry blocks for an activated message ... 3-138
3.11.1.2 Message entry blocks for an activated and a de-activated message ... 3-139
3.11.2 Configuring example for a message system ... 3-139
3.11.3 Output formats of the message evaluation block MSI ... 3-143
3.11.3.1 Structure of an error- or alarm message ... 3-143
3.11.3.2 Overview of the message formats ... 3-143
3.11.3.3 Structure of an overflow message ... 3-145
3.11.3.4 Structure of a communications error message ... 3-145
3.11.3.5 System error message structure ... 3-145
3.11.3.6 Detailed description of the message formats of function block MSI ... 3-146
3.11.3.7 Output format of the message evaluation block MSIPRI ... 3-151
3.12 Communications utility process data ... 3-153
3.12.1 Receive- and transmit blocks ... 3-153
3.12.1.1 Virtual connections ... 3-153

3.12.1.2 I/O of the CRV, CTV blocks ... 3-157
3.12.2 Channel marshalling blocks CCC4 and CDC4 ... 3-157
3.12.2.1 Group block CCC4 ... 3-157
3.12.2.2 Distribution block CDC4 ... 3-158
3.12.2.3 Compatible net data structure ... 3-159
3.12.3 Diagnostic outputs ... 3-159
3.12.3.1 Fault/error cause ... 3-159
3.12.3.2 Channel assignment ... 3-160
3.12.3.3 Channel statuses ... 3-161
3.12.4 Introduction – "Pointer-based communication blocks" ... 3-161
3.12.4.1 Principle mode of operation ... 3-162
3.12.4.2 Applications ... 3-162
3.12.4.3 Features of pointer-based communications ... 3-162
3.12.4.4 Associated function blocks ... 3-163
3.12.4.5 Pointer interface ... 3-164
3.12.4.6 Configuring information and instructions ... 3-164
3.12.4.7 Examples of CFC screenshots ... 3-166
3.13 Communications utility service ... 3-171
3.13.1 Function block SER ... 3-172
3.13.2 System load, response times ... 3-173
3.14 Communications utility time of day synchronization ... 3-174
3.15 Communications with SIMATIC Operator Panels ... 3-176
3.15.1 Configuring example ... 3-176
3.15.2 Configuring SIMATIC TDC ... 3-177
3.15.2.1 Selecting the components in HWConfig ... 3-177
3.15.2.2 Configuring with CFC ... 3-177
3.15.2.2.1 Initializing the OP7 ... 3-178
3.15.2.2.2 Reading function block connections (I/O) ... 3-178
3.15.2.2.3 Writing function block connections ... 3-179
3.15.2.2.4 Configuring events ... 3-179
3.15.2.2.5 Configuring alarm messages ... 3-180
3.15.2.2.6 Configuring the function keyboard ... 3-180
3.15.2.2.7 Configuring the interface area ... 3-181
3.15.2.3 Importing the symbol table ... 3-181
3.15.3 Configuring the OP7 with ProTool/Lite ... 3-182
3.15.4 Application information ... 3-183
3.15.4.1 Computation times ... 3-183
3.16 WinCC connection to SIMATIC TDC via the standard channel (SIMATIC S7 Protocol Suite.CHN) ... 3-184
3.16.1 Coupling via TCP/IP with "OCM" functions ... 3-185
3.16.1.1 Configuring the coupling-relevant TDC hardware ... 3-185
3.16.1.2 Configuring the CFC ... 3-186
3.16.1.2.1 Configuring the coupling-relevant CFC function blocks ... 3-186
3.16.1.2.2 Marking the function block connections in the CFC charts and creating the address book ... 3-188
3.16.1.3 Configuring WinCC ... 3-194
3.16.2 "S7DB" configuration version ... 3-199
3.16.3 MPI and PROFIBUS DP coupling versions ... 3-201
3.16.3.1 Hardware configuration ... 3-201
3.16.3.2 Configuring the CFC ... 3-206

3.16.3.3 Configuring WinCC ... 3-206
3.16.4 Configuring using the "D7-SYS-OS engineering tool" ... 3-208
3.17 Communications with WinCC (TCP/IP) ... 3-216
3.17.1 Prerequisites ... 3-216
3.17.2 Process variables ... 3-217
3.17.2.1 SIMATIC TDC software ... 3-217
3.17.2.2 Configuring WinCC ... 3-219
3.17.3 Binary events ... 3-220
3.17.4 SIMATIC TDC messages ... 3-220
3.17.4.1 SIMATIC TDC configuring software ... 3-220
3.17.4.2 WinCC configuring software ... 3-221
3.17.5 Generating the address book using the CFC editor ... 3-222
3.17.6 Address list import tool ADRIMP ... 3-222
3.17.6.1 Prerequisites ... 3-222
3.17.6.1.1 Generating the variable definition file ... 3-222
3.17.6.1.2 Generating and importing a new signal list ... 3-223
3.17.6.1.3 Importing an existing signal list ... 3-223
3.17.6.2 Checking the generated tag management in WinCC ... 3-224
3.17.7 Communications set-up, SIMATIC TDC-WinCC ... 3-224
3.17.7.1 Activating WinCC ... 3-224
3.17.7.2 Activating SIMATIC TDC ... 3-225
3.18 Communications service Trace ... 3-226
3.18.1 Simple Trace ... 3-226
3.18.1.1 Method of Operation of the @TCP ... 3-226
3.18.1.2 Method of Operation of the Acquisition Blocks ... 3-228
3.18.1.3 Method of Operation of the Header Block TRHI ... 3-229
3.18.1.4 Simple Trace Configuring ... 3-230
3.18.1.5 Reply Telegrams ... 3-232

Index ... I-1

1 In just a few steps to the first project

Section overview

1.1	Prerequisites	1-2
1.2	Creating a new project	1-5
1.3	Defining the hardware	1-5
1.4	Generating a CFC chart	1-6
1.5	Testing, compiling and downloading the project	1-10
1.6	Testing the user project	1-12
1.7	Results	1-14
1.8	Archiving the project	1-14

1.1 Prerequisites

Introduction

These brief instructions are intended for introductory level personnel and it outlines the basic procedure when generating a project.

More detailed information about the dialog boxes of the development software and their processing is provided in the corresponding online help.

1.1.1 Software and hardware

Software

Three software packages

- STEP 7
- CFC
- D7-SYS

must be installed precisely in this sequence on your PG/PC with Windows 95/98/ME/NT 4.0/2000. Authorization is required for STEP7 and CFC.

NOTE

The installation and user instructions are provided in the particular "readme" files. Please observe the interdependencies between versions!

When installing STEP7, you will be prompted for the online interface, however, for SIMATIC TDC nothing has to be selected and installed. ("Close" window and exit the following window with "OK".)

Hardware

You will require the following hardware components for the "My First Project" project example:

Components	Function	Diagram/Order No.
UR5213 subrack with power supply *21 slots, 32-bit bus, fan, 115/230 V AC*	... if the subrack is for a SIMATIC TDC station. ... it is used to mechanically accommodate the modules and supply them with power.	 **6DD1682-0CH0**
CPU module CPU551 *(at slot 1)* *64-bit, 266 MHz, 32 Mbyte SD-RAM, VME-bus, PCI-bus, 8 digital inputs, of which 4 are interrupt-capable*	... executes the user program. ... exchanges data with other modules via the backplane PC board of the subrack. ... communicates with a PG/PC via the serial interface.	 **6DD1600-0BA1**

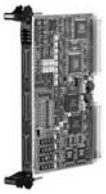
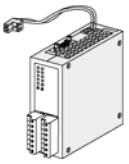
MC521 program memory module *2 Mbyte user program memory, 8 kbyte change memory*	... saves the operating system, the user program and the online changes.	 **6DD1610-0AH3**
PC cable SC 67	... connects the CPU module to the PG/PC.	 **6DD1684-0GH0**
SM500 signal module *(at slot 2)* *Signal modules, 16 DO, 16 DI, 8AI, 4AI integrating, 8AO, 4 pulse encoder inputs, 4 absolute value encoder inputs*	... expands the CPU module by technology-specific functions. It is especially fast, as it is directly screwed to the CPU module and the backplane bus is not used.	 **6DD1640-0AH0**
Interface cable SC 54 *Length: 2 m*	... connects the inputs/outputs of the SM500 signal module with up to 5 SBxx or SU12 interface modules.	 **6DD1684-0FE0**
Interface module SB10 *2 x 8 screw terminals, LED displays*	... allows you to test the user program during commissioning and in operation, as the statuses of the digital outputs are displayed using LEDs.	 **6DD1681-0AE2**

Fig. 1-1 Module list for the project example "My First Project"

NOTE Technical data is provided in the SIMATIC TDC Hardware Manual, additional ordering information in Catalog DA99.

1.1.2 What you can expect

From the task to the first project

The example "My First Project" guides you step-by-step to a project which can actually run.

1. **Analyze the particular task**
   This allows you to identify the function blocks, inputs and outputs which you require and which hardware:
2. **Define the hardware**
   You will use this hardware information in STEP7 in order to enter the modules and define your particular properties.
3. **Configure and compile**
   You generate the configured software in CFC using the function blocks and compile this. You can configure the hardware after all of the checks have been made.
4. **Test the configuring software**
   You can now run the program, tested online and change it on the SIMATIC TDC modules.
5. **Archive the project**

You can subsequently apply this procedure for you own applications.

The task

The task comprises two sections:

1. A **sawtooth generator** with a fixed frequency, outputs its value via a D/A converter.
2. **Running Lights** with 8 channels.

To start off with, define the individual functions for the appropriate sub-tasks and define the necessary hardware:

1. Sawtooth generator

A sawtooth waveform is generated by an integrator, which resets itself after an upper limit has been exceeded. The integrator value is output via an analog output.

2. Running light

Eight comparators compare the sawtooth value with constant values. The results are output through digital outputs and control the LEDs on the interface module.

The running light has the following phases:

- All of the LEDs are dark.
- The LEDs are switched bright and then dark again so that only one is bright at any one time.

1.2 Creating a new project

Step	Procedure	Result
1	**Double-click** on the symbol  . (if the STEP 7 Assistant starts, cancel this.)	The SIMATIC Manager is opened.
2	Select **File > New**. Enter **"My First Project“** into the dialog box, Project. In the dialog box, select the path **„LW:\Siemens\Step7\S7proj“**. Click on **OK**.	Your new project is displayed.
3	Select **Insert > Station > SIMATIC TDC station**.	The "**SIMATIC TDC station“** hardware object is inserted.

1.3 Defining the hardware

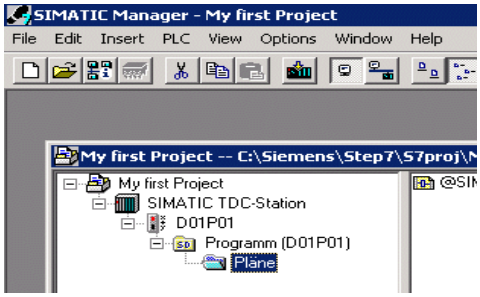
The SIMATIC TDC subrack structure is entered in STEP 7 (HW Config).

Step	Procedure	Result
4	Select the hardware object **"SIMATIC TDC station“** and select **Edit > Open object**.	**HW Config** is called-up.
5	Open it, if required, the hardware catalog with **View > Catalog**.	The hardware catalog with all of the available family of modules is opened.
6	Select the **UR5213** from the **SIMATIC TDC** family of modules and Catalog **Subracks** and drag it to the (upper) window	The subrack is displayed with 21 slots.
7	Locate them one after the other **>CPU Modules > CPU551** at slot 1 **>Signal modules> SM500** at slot 2 **>Slot covers > UR5213** at slots 3 to 21	The subrack is equipped.
8	Open the properties dialog box of the CPU551 CPU module with **Edit > Object properties**.	The CPU551 dialog box with general module information and the setting registers for addresses, basic clock cycle, cyclic tasks and interrupt tasks are displayed.
9	Select the basic sampling time T0 (in this case: 1 ms) under the **basic clock cycle** tab. Click on the **cyclic tasks** tab and set the sampling time T1 to 2 ms and T2 to 4 ms. Click on **OK**.	The required sampling times are entered. The properties dialog box is closed.
10	Open the properties dialog box of SM500 signal module using **Edit > Object properties**.	The SM500 dialog box with general module information and the setting tab for addresses is displayed.

11	Under the **Addresses** tab, click on the **Pre-assign** button. Click on **OK**.	All of the addresses are assigned symbolic names for subsequent use in CFC charts.
12	Check your hardware with **Station > Check consistency**.	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 has been fully configured.

1.4 Generating a CFC chart

1.4.1 Generating a new chart

Step	Procedure	Result
14	Change into the SIMATIC Manager and open the project tree up to the Charts object. Select the charts by clicking on them.	
15	Generate a new CFC chart twice with **Insert > S7 software > CFC** .	The CFC 1 and CFC 2 charts are displayed as new objects at the righthand side of the project window.
16	Select chart CFC2 in the project window and open the properties dialog box with **Edit > Object properties** . Enter the "sawtooth generator" name. Click on **OK**.	You obtain the properties dialog box of the CNC chart. The Properties dialog box is closed.
17	Repeat step 16 with the CFC2 chart and re-name it "Running lights".	The charts appear in the project window under their new name.

1.4.2 Inserting, parameterizing and inter-connecting function blocks

Step	Procedure	Result
18	Select the "sawtooth generator“ chart and open the "CFC Editor with **Edit > Open object**.	The CFC Editor is opened with the working area (>1 sheet) and the block catalog. (Catalog missing? Select **View > Catalog**) (>1 Sheet? Select **View > Sheet** view)
19	Open the family of blocks **Closed-loop control** and drag the function block **INT** (integrator) to the working area.	The block is now located on the sheet and has the ID for running in cyclic task T1.
20	Open the properties dialog box of function block INT with **Edit > Object properties** .	The INT dialog box with general block information and the setting tab I/O appears.

21	Under the **General** tab, change the name to „sawtooth“.	
22	Under the **I/O** tab, enter the values for the block inputs, e.g. • X = 1 • LU = 11250 • TI = 5 ms Click on **OK**.	The Properties dialog box is closed and the function block inputs now have values assigned.
23	First **click on** output QU and then on input S.	The output QU (upper limit) is now coupled back to input S (set).
24	Select **DAC** (analog output) from the block family **ON/OFF** and locate it next to function block INT. Open the dialog box using **Edit > Object properties** and change the name to "analog output“. Enter, for example under the **I/O** tab: • DM = 0 • OFF= 0 • SF = 1E6 Click on **OK**. Select connection AD (hardware address), and call-up the dialog box to interconnect the object with **Insert > Connect to operand**. Then mark the selection window. Select the first entry and click on **OK**	The block inputs are parameterized. The hardware address of the first analog output channel is assigned.
25	In the "sawtooth“ block, **click** on output Y and after this on input X in the "analog output" block.	The sawtooth generator is connected to the analog output.

All changes made in the CFC chart are immediately saved.

Proceed the same for the second sub-task (running lights) (from step 18). Change into the SIMATIC Manager, open the CFC chart "running lights" insert the function blocks into the CFC chart, parameterize and connect them.
All of the necessary information (number of blocks, types and block parameters) can be taken from the following diagrams. Arrange the first function block and all others, via
Edit > Run sequence in cyclic task T2. The connection between the "sawtooth" block and the comparators is realized by changing the CFC window (**Window > ...**).

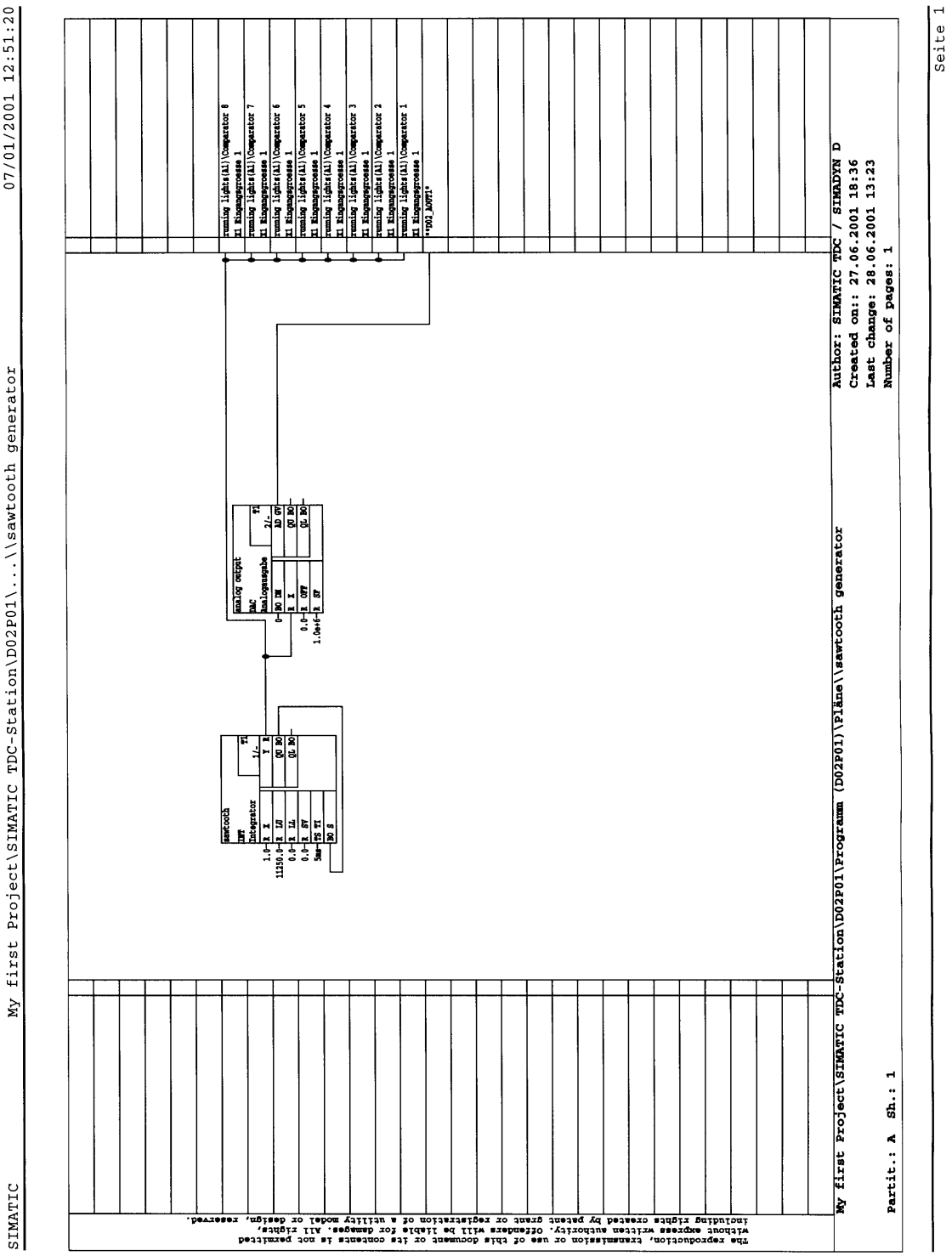


Fig. 1-2 "Sawtooth generator“ chart

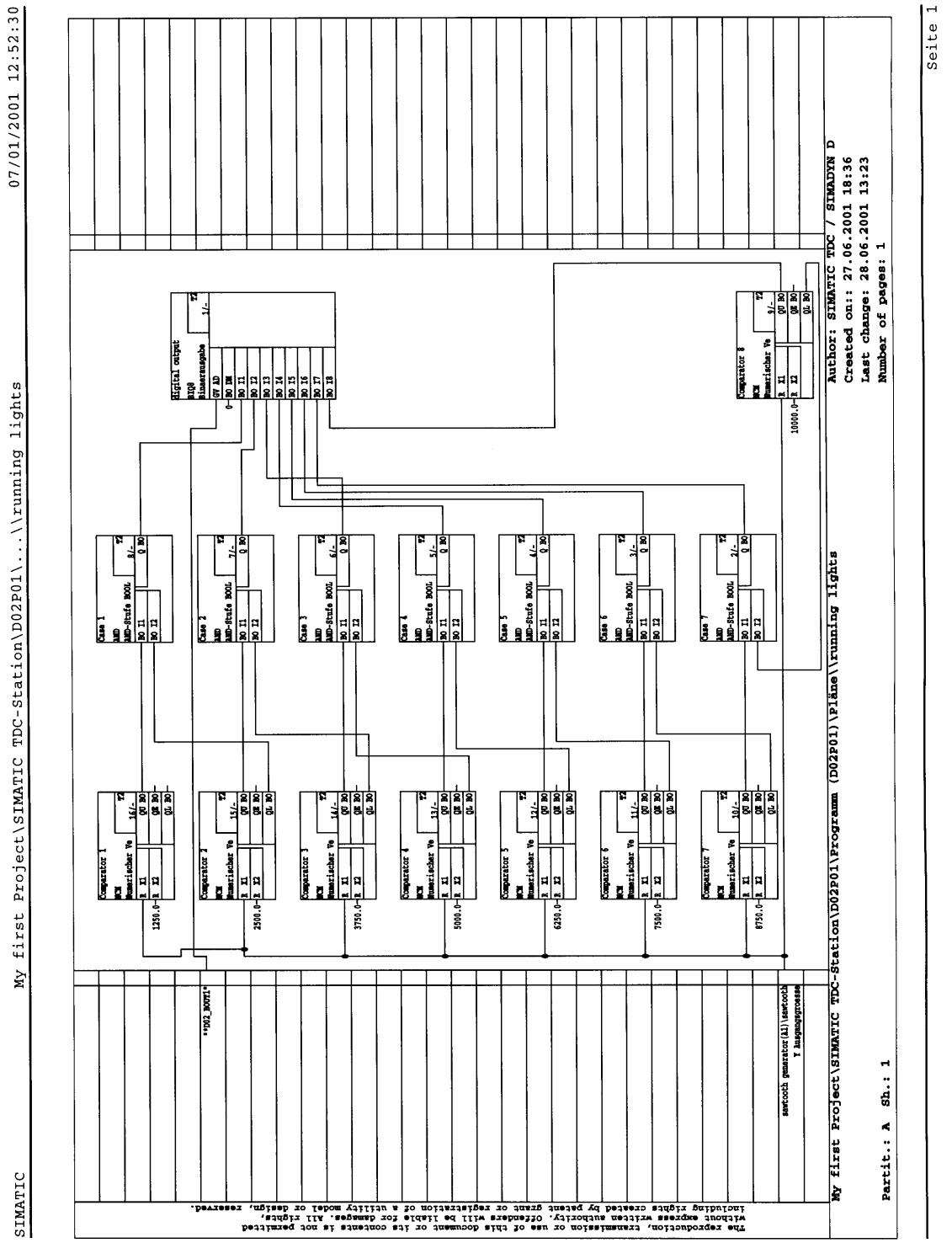


Fig. 1-3 "Running lights“ chart

1.5 Testing, compiling and downloading the project

1.5.1 Checking the project consistency and compiling

Step	Procedure	Result
26	Start the consistency check of your project with **Chart > Check consistency > Charts as program**, then **OK**. **Acknowledge** the dialog window or evaluate the error messages via **Details**.	The result is displayed in the dialog window.
27	Start to compile the project after a successful consistency check with **Chart > Compile > Charts as program**, then **OK**. **Acknowledge** the dialog window or evaluate the error messages using **Details**.	The result is displayed in a dialog window. You have created your first user project.

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

Introduction

SIMATIC TDC allows you to

- download online or
- offline.

Downloading offline

Maybe you do not have a connection from your PC/PG to the SIMATIC TDC station, which is why you can use the possibility of downloading into a memory module.

Step	Procedure	Result
28	Select **Target system > Download**.	You will obtain a dialog window with options.
29	Select "User program" and "Offline" Insert the memory module into the PCMCIA slot of the PG/PC. Start to download with **OK**.	A progress display shows how the system and your user program are being downloaded into the memory module.
30	Insert the memory module into the SIMATIC TDC station and re-start it.	Your user program is then started.

Downloading online

You have established a connection from your PC/PG to SIMATIC TDC station, and you can download the program memory module into the CPU module.

Step	Procedure	Result
28	Check whether your SIMATIC TDC station (hardware) is correctly configured, assembled and connected.	Observe the configuration instructions and connection possibilities for the individual hardware components in the appropriate hardware documentation!
29	Insert the memory module into the CPU module and start the SIMATIC TDC station.	A flashing zero appears on the CPU module display
30	Install the interface between the SIMATIC TDC station and the PC in the SIMATIC Manager using the menu command: **Options > Set PG/PC interface...**.	You obtain a dialog window "Install/uninstall interfaces" in which the various interfaces are listed.
31	In the dialog window, select "DUST1 protocol" and install this protocol with **Install→** Acknowledge with "yes" and then **close** the dialog window. Select the interface used and acknowledge with "**OK**".	You obtain a dialog window in which you can decide, by entering either "Yes" or "No" whether you wish to immediately go online. The "Set PG interface dialog window" is displayed where you can select the access route "DUST1 (COM1)" or "DUST1 (COM2)".
32	Select the **Target system > Download**.	You obtain the dialog window with options.
33	Select the "System and user program", "Online (COM1)" and initialization when first downloading the user program. **Note:** If a user program is downloaded again, you can also specify "User program" without "initialization". Start with "download"	A progress display shows how the system and your user program are being downloaded into the memory module. If download has been completed, the dialog window "Operating status" is displayed with the "STOP" status.
34	Start the SIMATIC TDC station with "Restart" and then select "Close".	Your user program is started and the "Operating status" dialog window is displayed with the "RUN" status.

1.6 Testing the user project

Introduction

In the test mode, you can

- Monitor the values of block I/O and change the values of block inputs,
- Generate and delete connections, and
- Insert and delete blocks.

The values which are registered for test, have a yellow background. You can easily monitor the behavior by changing parameters at the block inputs.

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

- You have established a connection between the PG/PC and your SIMATIC TDC station.
- You have downloaded the actual project into the memory module, which is located in the CPU module.
- The associated CFC chart (e.g. "running lights") has been opened.

Step	Procedure	Result
35	Select the menu command: **Target system > Compare**, to display the "Compare" dialog field.	The CPU name with data and time of the last compilation between the actual configured software and the current CPU program are displayed. If they match, the result is: "The configuring and the CPU program match". You have checked that the PG/PC and the SIMATIC TDC station can communicate.
36	Select the menu command: **Test > Test settings** Enter the refresh period for the screen display in tenths of seconds. Acknowledge the change with "OK".	In the test mode, the values of the I/O are updated cyclically on the screen with the selected refresh period. If the computation time is not sufficient to fulfill the refresh periods, then you will be warned. The closed-loop control always has the higher priority
37	Before you go into the test mode, change over the test mode from "Process operation" to "laboratory operation" with **Test > Laboratory operation**. **Note:** In "Process operation", the default setting is that **no** I/O are registered for monitoring. In this test mode, you must select the appropriate blocks and explicitly log them-on for monitoring.	This means that all of the block I/O are automatically switched-in for "monitoring" (the values have a yellow background).
38	Select the menu command: **Test > Test mode**	The "Test: RUN (laboratory)" text appears with a green background in the status bar. In the test mode, you can monitor and change the dynamic behavior (online).

1.6.1 Disconnecting the connection online

Procedure In the CFC chart, using the mouse pointing device, select the block I/O which you wish to disconnect. Then remove this with **Edit > Delete**.

Result The connecting line between the I/O disappears and at the I/O, the last value, which was transferred on the connection, is displayed as parameter value.

NOTE Connections to global operands can neither be generated online nor deleted.

1.6.2 Generating a connection online

Procedure In the CFC chart, using the mouse pointing device, select the block I/O where you wish to establish a connection.
With the changeover key pressed, now select the block I/O to which this connection should be made.

Result The connecting line between the selected I/O is generated, and the actual parameter value, which is presently being transferred, is displayed at the output.

1.6.3 Changing the parameterization online

Procedure Select the block input whose parameter value is to be changed, by double-clicking. The dialog box "Properties I/O" is displayed in which you can change the value.

Result You can immediately identify the effect of the change in the CFC Chart

1.6.4 Inserting a block online

Procedure Using the command **View > Catalog**, call-up the block catalog. Open the block family and drag the selected function block to the working area.

NOTE Not all of the function blocks can be inserted online. Refer under "configuring data" in the online help for the block.

1.6.5 Deleting blocks online

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

1.7 Results

You have now got to know some of the simple handling operations in the CFC configuring. You now know how a project is created using the SIMATIC Manager, how a CFC Chart is generated and function blocks inserted from a library. You have interconnected and parameterized the function blocks. You have generated a program which can run and which has been downloaded into the CPU. You can observe and modify the dynamic behavior in the test mode

You can now review the results for the project example "My First Project“ in **process operation** if you have assembled and connected-up the necessary hardware of the SIMATIC TDC station (refer to Table 1-1, Section 1.1.2).

Sawtooth generator

In order to view the sawtooth, you must first connect an oscilloscope to the SIMATIC TDC station. The following table shows the assignment of the pins at output connector X1 of SM500 signal module. The output voltage range extends from -10 V to +10 V.

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

Table 1-1 Excerpt from the pin assignment of SM500, connector X1

Running light

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

1.8 Archiving the project

Step	Procedure	Result
44	In the SIMATIC Manager, select **File > Archive**.	The "archiving" dialog field is displayed.
45	In the dialog field "Archiving", select the user project with "My First Project“. Click on **OK**.	The "archiving - select archive dialog field" is displayed. The default file "My_first.zip" has already been entered with archiving path.
46	In the dialog field "archiving - select archive", when required, change the file name and/or the path and then click on "save"	The project is now saved in the selected path and filenames as zip file.

NOTE When you select menu bar **File > De-archive**, the archived project can always be re-established with this particular release.

2 Systemsoftware

Overview

2.1	Configuring	2-2
2.2	Function description and user instructions	2-40
2.3	System chart @SIMD	2-44

2.1 Configuring

2.1.1 General description

This Chapter provides instructions and support when configuring SIMADYN D. It explains the general requirements when configuring SIMADYN D hardware and software.

It is assumed that the reader is knowledgeable about Windows 95/98/NT, handling the SIMATIC Manager, HWConfig and the CFC Editor; they will not be explained in this document. The configuring instructions are illustrated using diagrams and graphics. These illustrations are intended to highlight specific features, and do not necessarily precisely illustrate the CFC window. This Manual does not discuss the hardware (e. g. CPUs, memory modules, cables etc.), even if hardware designations are used in the configuring examples; if hardware information is required, then please consult the "Hardware" User Manual.

This Manual is sub-divided into the following Chapters:

- General description
- Configuring the hardware
- Creating CFC charts
- Operating statuses of a CPU module
- Configuring example for a CPU module
- Using signal transfer mechanisms
- Significance and uses of the process image
- Significance and uses of the CPU synchronization
- Significance of processor utilization

To implement most of the applications, the information in Chapter "General description" up to the Chapter "Creating CFC charts" is sufficient. More detailed information regarding special system characteristics of SIMATIC TDC/SIMADYN D is described in the following Chapters.

2.1.1.1 Configuring tools

In practice, a configuring engineer can select the required hardware modules from a module spectrum and achieve the desired technological functions by generating function diagrams and block diagrams. SIMATIC TDC/SIMADYN D supports these activities using ***HWConfig*** (configuring tool to define the hardware configuration of SIMATIC TDC/SIMADYN D stations) and ***CFC*** (block technology using numerous standard function blocks).

2.1.1.2 Configuring steps

SIMATIC TDC/SIMADYN D is configured in the following sequence

1. The hardware configuration is generated, and
2. The CFC charts are created.

2.1.1.3 Terminology and libraries

Assigning a name

When configuring SIMATIC TDC/SIMADYN D, the names to be assigned must be as follows:

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

Sequence	Characters permitted	Example
First character	Alpha- and special characters	A-Z, @
Second character	Alphanumeric characters and special characters	A-Z, 0-9 , _ , or @ if the first character is @
Additional characters	Alphanumeric characters and special characters	A-Z, 0-9 , _

Table 2-1 Nomenclature when assigning names to modules

- Chart- and function block names
  - when both names are connected, the total number of characters may not exceed 24.

Name	Max. length	Permitted characters	Characters which are not permitted
Chart	22		*, _, ?, <, >, |
Function block	16		“

Table 2-2 Nomenclature when assigning names to charts and function blocks

- Comments
  - for modules, maximum of 255 characters
  - for charts, maximum 255 characters
  - for function blocks and parameters, max. 80 characters

- Connections (I/O) with special functions have the following suffixes:
  - the dollar symbol "$" (connecting signals between CPUs),
  - the star symbol "*" (symbolic hardware addresses),
  - or the exclamation mark "!" (virtual addressing).

HWConfig or CFC automatically enter these suffixes. A function block name may only appear once on a CPU. The name syntax and rules are checked when entered.

Libraries

Hardware modules and function block types are saved in libraries. The required function blocks can be called-up from the libraries using HWConfig or the CFC editor.

Several function block libraries can be used for each CPU. The "FBSLIB" standard function block library is pre-assigned. It has over 200 function blocks, whose functionality is sufficient for most applications. When required, additional supplementary libraries can be imported for the particular CPU. The libraries can be found in the directory "step7\s7cfc\sdblocks\std (SIMADYN D) or ...\tdc (SIMATIC TDC)".

2.1.2 Configuring the hardware

Configuring SIMADYN D stations

HWConfig is used to configure the hardware of SIMATIC TDC/SIMADYN D stations. A SIMATIC TDC/SIMADYN D station consists of a rack with up to 20/8 CPUs and other hardware modules. When required, several stations can be coupled with one another. The modules to be configured can be selected from the modules in the HWConfig hardware catalog. Racks, CPUs, I/O modules, coupling modules etc. can be selected.

HWConfig defines the system hardware configuration as result of

- the rack used together with the defined bus structure (bus termination, Daisy Chain),
- the configured hardware modules inserted in the rack as well as
- defining hardware-relevant information such as tasks, synchronization etc.

2.1.2.1 The first step: Selecting the hardware modules

Short overview of the hardware

The following modules are available in the HWConfig hardware catalog:

Hardware	Description
Subracks	Various types depending on the slot number, bus configuration, cooling etc.
I/O modules	Peripheral modules to input/output process signals (analog-binary I/O, speed sensing signals etc.)
Expansion modules	Peripheral modules to input/output process signals. They are used to achieve higher data rates by bypassing the backplane bus, and are directly connected to a CPU module.

Hardware	Description
Communication modules	Modules to provide communication utilities
Communication buffer modules	Modules to transfer data between several CPUs
CPU modules	Modules on which the configured open-loop or closed-loop control program is executed. A maximum of two expansion modules can be inserted next to a CPU.
Special modules	Modules with special functions.
Slot covers	Slot covers cover empty slots against dirt accumulation and as EMC measure
Sub-modules	A sub-module is inserted in or on a module, e. g. a memory module for a CPU or an interface module for a communications module
Technology components	Subracks as well as modules for drive converters

Table 2-3 Hardware components

Further information
Refer to the " SIMATIC TDC/SIMADYN D hardware“ Manual for the individual modules which can be selected.

Using HWConfig, a module is configured, possibly with a sub-module for every subrack slot. This provides a precise image of the rack as it is in reality while the hardware is being configured. When selected, each module is given a name (recommended) which can be changed in accordance with the syntax for names. Slot covers must be provided for those slots which remain empty.

2.1.2.2 The second step: Parameterizing the hardware modules

After they have been selected, the modules must be parameterized using HWConfig. The following must be set

- the sampling times of the cyclic tasks,
- synchronizing cyclic or interrupt-control tasks of several CPUs of a station,
- the process interrupts and comments

Various parameterizing dialog windows are provided in HWConfig for this purpose.

Parameterizing dialogs in HWConfig

The pre-settings of the modules can still be changed in the module dialog windows. For instance, the parameterizing dialog for CPU modules includes the "Cyclic tasks" information. This allows the sampling times of 5 cyclic tasks to be changed.

Designation schematic

At least one rack and all of the modules and sub-modules which it accommodates must be configured in HWConfig. When a module is generated, a recommended module name is assigned. This recommended name can be overwritten as long as it conforms to the maximum name length (max. 6 characters) and the character exists (refer to the Chapter "General description"), with (A-Z,0-9,_,@). It is recommended that the names are selected according to the schematic in the following table for the plant/system components:

Hardware	Logical name	Designator	Significance
Subrack	An00	n	Subrack number, starting at 1
CPU	Dxy_Pn	xy n	Slot number CPU number
Sub-module	Dxyj	xy j	Slot number Sub-module number
Communication buffer 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

Table 2-4 Designation schematic for the hardware configuration in HWConfig

Slot number definition

The slot number of a module specifies the number of the slot in the subrack where the actual module is configured. For a SR24 with 24 slots, these are slots 1 to 24.

All sub-modules of a module are consecutively numbered starting from 1. The sub-module which is located at the top of the table is number 1.

The recommended CPU rack name is 6 characters long. The logical processor number (in the rack, from left to right) is displayed in operation, independently of the assigned name on the 7-segment display of the CPU module.

NOTE The configured module names within a station must be unique.

The various tasks of a SIMADYN D CPU

The configured function blocks are processed via

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

The start of an interrupt task with respect to the instant that the process interrupt was initiated can be offset by a freely-configurable delay time.

SIMADYN D system chart

The system chart, in which the behavior/characteristics of the 7-segment display, acknowledge button etc. is configured, is administered in a newly created SIMATIC TDC/SIMADYN D program, and may not be deleted. The sampling time of the system chart is pre-assigned in the factory at approx. 128 ms.

2.1.2.3 The third step: Checking the configuring

When the hardware configuration has been completed, the configured data must be verified using a consistency check over the complete station. The complete hardware configuration is checked using HWConfig. If the software has bugs or is incomplete, these are displayed and can be „debugged“ (refer to the Chapter "Configuring example of a CPU module").

2.1.3 Creating CFC charts

Description of the CFC editor

A CFC chart (Continuous Function Chart) is generated using the CFC editor. This is a configuring tool to describe continuous processes by graphically interconnecting complex functions in the form of individual function blocks. Thus, the CFC is used to graphically implement a technological application by interconnecting and parameterizing function blocks. For a configuring engineer this means that he can program using a system which is closely related to block diagrams.

CFC chart structure

A CFC comprises of several CFC charts, each with 6 sheets. Each sheet can have a different number of various function blocks. The actual number is only limited by the graphic layout. In the overview of the CFC editor, all 6 sheets of a chart are displayed, and in the sheet view, an individual sheet can be displayed in detail. The function blocks which can be called-up in the CFC editor are sub-divided into function block classes, which include the interconnected (associated) functional scope. For instance, this can include logic blocks, arithmetic blocks etc.. Each function block class in turn includes a number of various function block types.

The CFC editor defines the technological configuring by:

- selecting, interconnecting and parameterizing the configured function blocks,
- defining of the sequence characteristics of the function blocks,
- generating programs to program the CPU memory modules.

2.1.3.1 The first step: Selecting the function blocks

The various function block classes are available in the FBSLIB standard library. The individual function blocks can be called-up using the CFC editor, and located on the chart sheets. Individual blocks or block groups can be subsequently deleted, shifted and copied at any time.

Additional information
For further information on the function blocks refer to the Reference Manual "SIMADYN D function block library".

2.1.3.2 The second step: Parameterizing and interconnecting function blocks

After the function blocks have been selected, these are interconnected and parameterized using the CFC editor. The task, in which the individual function blocks are computed, must also be defined.

Parameterizing dialogs in the CFC editor

By double clicking on the function block header or under the menu selection **Edit > Object characteristics**, the following data can be configured deviating from the pre-settings:

Data	Description
General	The name and a comment text which is displayed in the function block header can be configured. Under "special object properties" you can execute the steps which are necessary to prepare a block for operator control and monitoring using WinCC.
Run-time properties	Here, the execution sequence of a function block, defined under function block insert, can be changed within a task. The selected function block can be "searched for", "cut-out" in the execution sequence, and "inserted" in another position.
I/O	The following I/O data can be entered here for all parameters: • value and comment for input and output parameters • visibility in the CFC chart for parameters which are not interconnected • set or inhibit parameter ID for test • scaling value for parameters, REAL data type • texts for the various units

Table 2-5 Configuring function blocks

Additional information
Refer to the Manual "D7-SYS - STEP 7, CFC and SFC configuring".

Defining the run-time properties

Function blocks which are consecutively executed within a task can be combined to form a run-time group. In addition to structuring the task, this allows task execution to be individually enabled/disabled.

NOTE

If a run-time group, is disabled via a function block input which is connected to it, then all of the function blocks contained in it are no longer computed.

By assigning the function blocks to a cyclic or interrupt-controlled task or run-time group and defining the position within the task or run-time group the configuring engineer can define the run-time properties of the function blocks. These properties are decisive for the characteristics of the target system as far as

- deadtimes,
- response times,
- the stability of time-dependent structures.

Assigning the function blocks to cyclic tasks

The function blocks are assigned to one of the 5 possible cyclic tasks by calling-up the block using the CFC editor or in the program section, execution sequence of the CFC editor. Each function block can therefore be assigned to a cyclic task and a processing sequence within the sampling time of the task.

Assignment of the function blocks to interrupt task

In order to process function blocks and run-time groups, interrupt-controlled, when they are called-up, or in the execution sequence of the CFC editor, they are entered in the required sequence under one of the 8 possible process interrupts. Thus, individual function blocks or a run-time group can be executed, initiated by a specific process interrupt.

NOTE

Contrary to cyclic tasks, interrupt tasks are not started in equidistant time intervals, but when a process interrupt occurs.

Configuring the equivalent sampling time

Several function blocks, e. g. some control blocks, have to be processed at regular interval as result of the program design. If these are to be configured in an interrupt task, then an equivalent sampling time must be configured in the HWConfig program section for this particular interrupt task. This should approximately correspond to the average time between two process interrupts.

By clicking twice on the module, you can configure the equivalent sampling time under the menu item **Basic clock > Synchronization**.

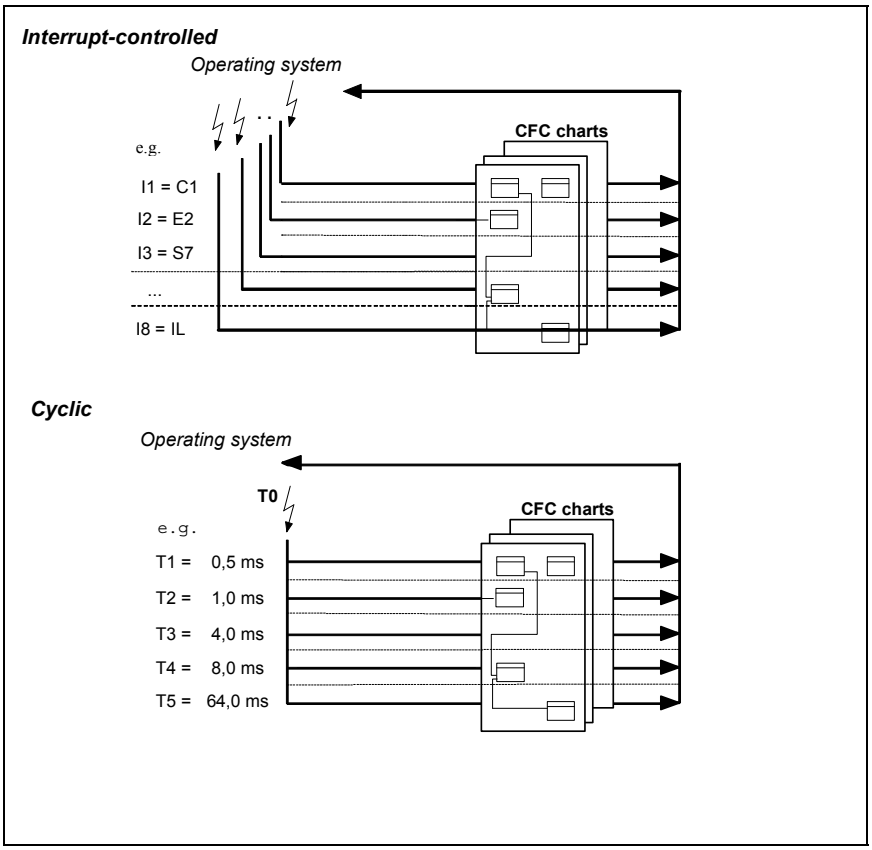


Fig. 2-1 Function block processing by the operating system

Executing the function blocks

The actual open-loop and closed-loop control task can be implemented using SIMATIC TDC/SIMADYN D, almost the same as in a block diagram, by interconnecting and parameterizing the function blocks. A function block type can be used as often as required. The function blocks are parameterized and interconnected at the block inputs and outputs.

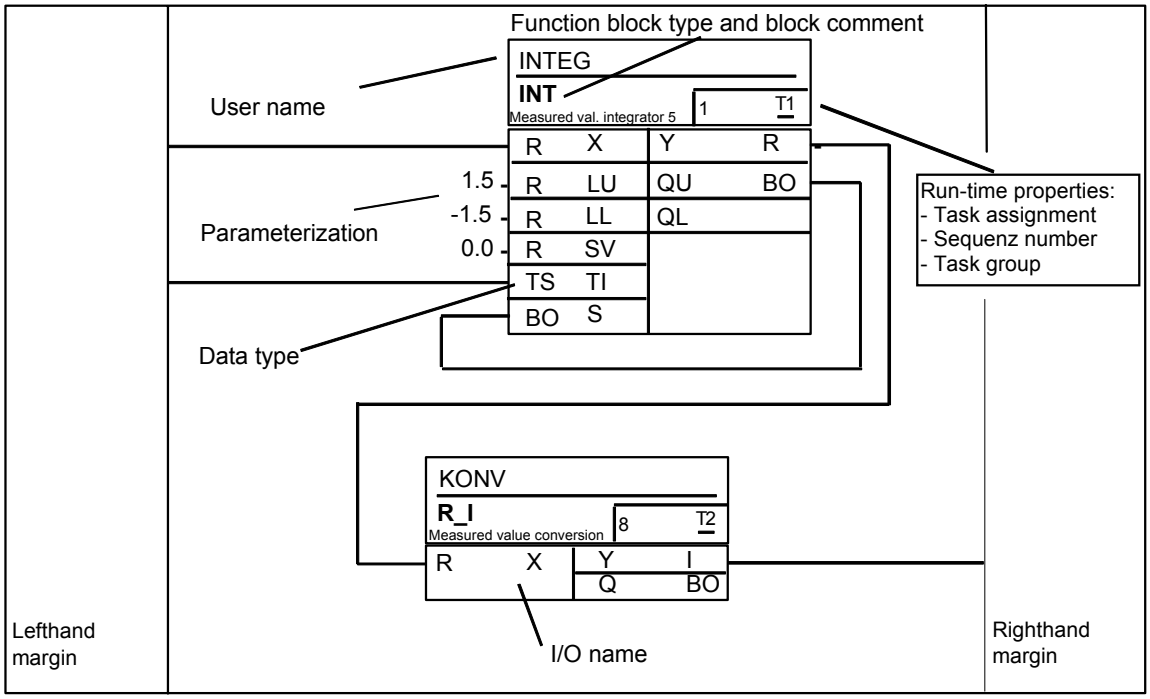


Fig. 2-2 CFC chart sheet- work area

For general parameterization of the function blocks and interconnections between the function blocks, there are

- inputs (function block inputs) and
- outputs (function block outputs).

Inputs

The configuring engineer can parameterize the inputs with constants or connect them to other function block outputs. When the function blocks are called-up, the inputs and outputs are pre-assigned, but these can be changed.

Outputs

The outputs can be connected to other inputs or assigned an initialization value which is different than the pre-assigned value. This value is available at this output if the function block is executed for the first time in the INIT operating status. This is practical, if the output of a flipflop block is to be pre-assigned.

Margins

The margins at the left and right of a CFC chart include, on one hand, the references to the objects to be interconnected, e. g. other blocks or run-time groups, which are not located on that sheet. On the other hand, they also include the number of the connector (termination location), if the autorouter cannot draw the connecting line to the margin as the sheet is overfilled.

Overflow sheets

Overflow sheets are automatically created, if more margin entries are generated on a sheet than there is space to display them. An overflow sheet consists exclusively of the margins and does not contain any objects.

Parameterization

Instead of an interconnection, a constant, deviating from the pre-assigned constant, can be parameterized at each input or output.

A block connection can be designated as parameter using a pseudocomment.
Additional information
on parameterizing, refer to the Manual "System and Communications Configuring D7-SYS , Section "Parameterizing SIMATIC TDC/SIMADYN D"

Interconnecting

Interconnecting involves the following:

- connecting a function block output to another function block input on the same CPU.
- connecting a function block output to a run-time group
- connecting a function block output to a global operand or a global operand with a function block input. A global operand can be:
  - a name with a "$" dollar symbol as suffix, i. e. connecting a signal from or to a function block on another CPU.
  - a virtual connection name or a virtual connection, i. e. transferring process data between function blocks or via any links using the process data utility.
  - a symbolic hardware address. A hardware address is in this case a symbolic designation of one or several associated terminals of a module. For example, binary inputs of a binary input module. The symbolic hardware address is defined in the HWConfig program section.
  - a name reference, i.e. the name of a message system

All types of interconnections which leave a chart sheet, generate an appropriate cross reference at the margin of the CFC chart.

Comments:

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

Pseudo comments

There are three pseudo comments, which are identified by the @ character as suffix and can be separated by blanks in front of the standard comment text:

1. @DATX

- The input is connected, bypassing the consistency mechanisms (refer to the Chapter "Description and use of the signal transfer mechanisms").

2. @TP_bnnn

- A connection identified like this can also be addressed as parameter. (The parameter can be read and changed at the block inputs using operating control devices and can only be read at the block outputs:

- The I/O, defined as parameters, can be read and changed via these interfaces; and also via drive converter operator control panels or SIMOVIS. The following variables are used:
- b: range identification "H", "L", "c" or "d"
  - identifies the parameter number range
  - "H" or "L": connections can be read and changed
  - "c" or "d": connections can only be read
- nnn: three-digit parameter number
  - 000 to 999

3. @TC_nnnn:

- A technology connector @TC_nnnn at a block output can be interconnected with a parameter at a block input using BICO technology. A technology connector is identified using ist number:
- nnnn: four-digit technology connector number
  - 0000 to 9999

Additional information
on parameters and technology connectors, refer to Manual "SIMADYN D Control System, Communications Configuring D7-SYS", Section Parameterizing SIMADYN D.

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

After all of the required hardware modules have been configured with HWConfig and the required function blocks on the individual charts using the CFC editor, the software can be compiled into the CPU machine code using the compiler. There are 2 ways to do this:

Offline loading A memory module is programmed with the PCMCIA interface of the configuring PC. After all of the correctly programmed memory modules of all of the subrack CPUs have been inserted, the modules are ready.

Online loading The user program and operating system are directly loaded from the configuring PC into the CPU via a serial communications link.

2.1.4 Operating statuses of a CPU module

In the SIMATIC TDC/SIMADYN D system, the system statuses, shown in the following table are possible:

Operating status	**Power off**	**I N I T**	**R U N**	**S T O P**		
Internal system status				**User stop**	**Initialization error**	**System error**
Status description	No-voltage condition	System run-up (initialization)	Cyclic operation (standard operation)	Stop initiated by the user	Status after initialization error	Status after fatal system error
Characteristics/properties	System not operational	System run-up --> external control not possible	Functionality in accordance with that configured	No cyclic processing -> fast download	Initialization erroneous --> no transition into cyclic operation	Fatal system error -> processing aborted
7-segment display	Dark	'0'	PN number ('1' ... '8') and 'C', 'E', 'b', 'A'	'd' (flashing when downloading)	'0' (the cause flashes)	'H' (the cause flashes)
Red LED on T400	Dark	Off	Flashes at a low frequency	Flashes at a medium frequency	Flashes at a high frequency	Lit (bright)
Available diagnostic interfaces	--	None	All of those configured (**one must be at the first CS7-SS**) and local interface		Local interface and first CS7 interface	Local interface
Possible operator control functions	--	None	Complete functionality of CFC online	Only diagnostics or download	Only diagnostics or download	Only diagnostics or download
Administered through the user interface (CFC)	--	--	The user can interrogate the statuses per interactive dialog			

Table 2-6 System statuses of a CPU module

Terminology

Term	Description
First CS7-SS	SIMATIC TDC: Interface which is inserted at the top in the first CP50M0/CP51M1 in the subrack (when counting from the left). SIMADYN D: Interface module (SS4 or SS52) which is inserted at the top in the first CS7 in the subrack (when counting from the left).
Diagnostics	Only possible to read-out error fields

Table 2-7 Terminology

INIT, RUN and STOP are the operating statuses, whereby STOP is subdivided into three different system statuses.

System status user stop

The "User stop" system status has been newly implemented and is used to **quickly** load a program via SS52/MPI, SS4/DUST1 (SIMADYN D), CP50M0/CP51M1 (SIMATIC TDC) or a local interface. Fast program loading means that cyclic processing is stopped in this status and the full performance of the CPU is available for download. A 'd' is displayed in the 7-segment display which starts to flash when a program is being loaded. This status is initiated by the user, whereby the parameterization as far as the configured diagnostics interface is concerned still remains valid (SS4, SS5x interface module in the CS7 module, CP50M0/CP51M1).

Download in the RUN status

It is also possible to download a program in the RUN status using each utility however this does not involve significantly longer download times (data is loaded in parallel with the cyclic processing).

It is only possible to changeover into the "User stop" status out of the RUN status by the user explicitly requesting this via a service interface (local or configured). In this status, all configured service interfaces and the local service interface are still available, i. e. diagnostics and downloading are still possible via all of the service interfaces (this is necessary if several PCs are connected at the rack).

2.1.5 Description and use of the signal transfer mechanisms

Signal transfer is data exchange between various blocks.

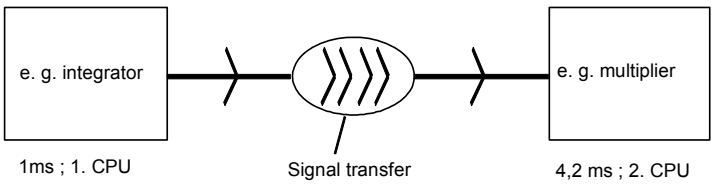


Fig. 2-3 Data transfer between two tasks

2.1.5.1 Data consistency

For interconnections between different cyclic tasks, SIMATIC TDC/SIMADYN D ensures the consistency of all data which is transferred. This means, that all data transferred from a task come from the same computation cycle of this task. All values calculated during a sampling cycle are "exported" at the end of the task. When starting a task, the required values are "imported", whereby it is ensured that there is no overlap (from a time perspective) between reading and writing the values (buffer system). As deadtimes are unavoidable with this concept, a signal should not be routed via several tasks and CPUs - if this can be avoided.

A differentiation is made between the following signal transfer types :

- Data transfer within the same task of a CPU
- Data transfer between various tasks of a CPU
- Data transfer between cyclic tasks of several CPUs
- Data transfer between alarm tasks of several CPUs

2.1.5.2 Data transfer within the same task of a CPU

Each function block output in the system is assigned a memory location. The function block saves its computed value in this memory location after being processed. All inputs, which are connected with the outputs in the same task, retrieve their values from the memory locations assigned to the connected output. In order to prevent deadtimes, the blocks of a task should if possible be computed corresponding to the "signal flow“, i. e. that block whose outputs are used as inputs for the following block is first computed etc.

2.1.5.3 Data transfer between various CPU tasks

Data transfer between various tasks of a CPU is realized via a buffer system so that the data consistency can be guaranteed (refer to the Chapter "Data consistency"). However, for data transfer from a faster to a slower task, it should be observed that value changes are not sensed in the slow task or are only sensed with a delay. If this cannot be tolerated, then the software must be appropriately adapted, e. g. using pulse-extending function blocks.

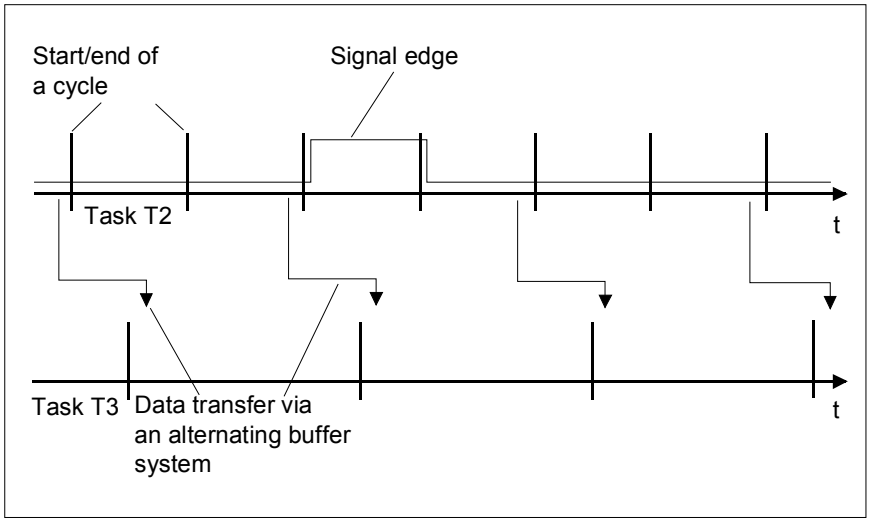


Fig. 2-4 Signal not sensed in task 3

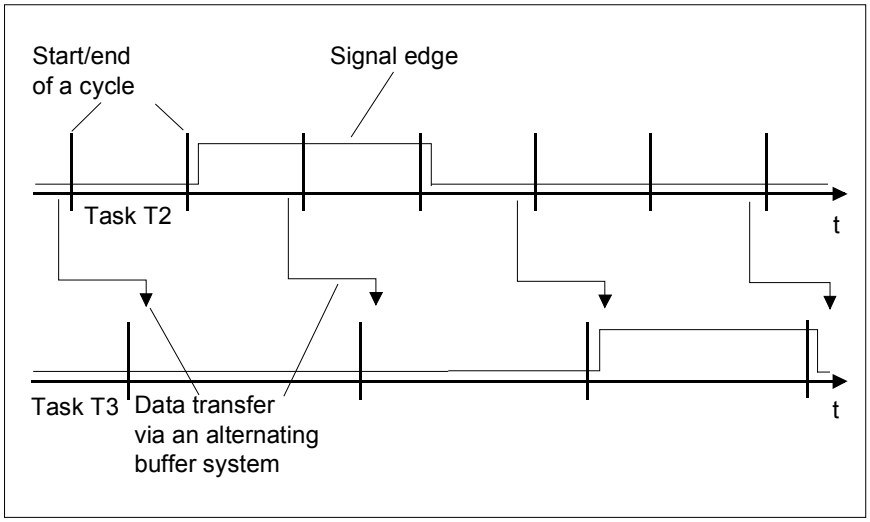


Fig. 2-5 Signal sensed with delay

2.1.5.4 Data transfer between cyclic tasks of several CPUs

Signals are transferred between the CPUs using the MM3, MM4 and MM11 (SIMADYN D) or CP50M0/CP51M1 (SIMATIC TDC) communication buffer modules. $ signals are used to handle the connections between function blocks, which run on different CPUs within the same SIMADYN D station (menu item "Insert-connection to the operand " in the CFC editor). The following data are required to configure a $ signal:

- the 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, the user (destination) can always access a current value. The deadtime, generated during signal transfer is then minimal if the generator (source) and user (destination) are configured in the same task, and if the tasks are possibly synchronized (refer to Chapter "Significance and application of the CPU synchronization").

The bus assignment defines whether data is to be transferred via the L bus or the C bus.

NOTE If time-critical functions are processed on the CPUs of a subrack, then please observe the following rules:

- Limit the number of $ signals to a minimum.
- Select the L bus for the $ signals, which are configured in interrupt tasks (alarm tasks).
- Select the C bus for the $ signals, which are not configured in interrupt tasks (alarm tasks).
- If possible, configure all of the communication links of the rack coupling to one or a maximum of two CPUs of the subrack.
- Configure the CPUs with the configured communication links of the rack coupling so that, if possible, there are no additional CPUs between these CPUs and the rack coupling module.

2.1.5.5 Data transfer 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. This is the because an interrupt event can occur at any instant in time and therefore the consistency mechanisms must be bypassed in order to prevent data loss. In this case, a conflict could occur between the demand for data consistency and low deadtimes. A decision must now be made depending on the particular application.

NOTE It should always be checked as to whether problems could occur if there is no data consistency (data consistency mechanism bypassed).

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

Time interval	Computation
Minimum value	1 * Tx
Maximum value	2 * Tx + 1 * Ty + 1 * T_interrupt

Table 2-8 Deadtime computation

- Tx = sampling time of the cyclic tasks through which the signals are looped,
- Ty = sampling time of the source/destination (target) CPU and
- T_alarm = maximum interrupt repeat time of the interrupt task.

2.1.5.6 Minimizing the deadtimes

To minimize the deadtimes, a signal can be directly transferred, bypassing the data consistency mechanism. It can be directly "connected" to the output of the generating block. They are two ways to configure this:

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

2.1.5.7 Processing sequence within a basic CPU clock cycle

The task administrator (refer to the Chapter "Mode of operation of the task administrator") of the operating system is started with the basic CPU clock cycle T0. This then decides which tasks are to be started (T1 and maximum of one other Tn,

with Tn from {T2...T5}.

Essentially, the following components are to be executed within the task processing:

- Buffer changeover for the tasks to be started (T1 and, if required an additional task Tn)
- System mode of the blocks in T1 corresponding to the module sequence (refer to the Chapter "Significance and uses of the process image")
- System mode of blocks in Tn corresponding to the block sequence (refer to the Chapter "Significance and uses of the process image");
- **Importing signal interconnections in the T1 and standard mode T1**
- **Exporting signal interconnections from T1**
- **Importing signal interconnections in Tn and standard mode Tn**
- **Exporting signal interconnections from Tn**

The components relevant for signal transfer are highlighted.

2.1.5.8 Interconnection changes and limited number of interconnections

Interconnection changes during the configuring test phase

Interconnections extending beyond the task limits can only be changed with some restrictions using the test mode of the CFC editor. The CFC editor test mode is used to test and optimize the user program, which is already running online on the CPU.

When service makes changes such as these, there are only a limited number of reserves for additional interconnections. The number of additional interconnections is

- minimum of 10 additional interconnections, and
- maximum of 20 % of the already configured number of interconnections.

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, as 20 % of 5 = 1, however a minimum of 10.

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 communication buffer modules. The number of possible interconnections is dependent on the communication buffer module used and the signal types.
Further information
on the communication buffer modules refer to the " SIMATIC TDC/SIMADYN D hardware" Manual

For an MM11 module with 64 Kbyte memory each for the L- and C bus, the following are obtained when using:

Signal type	Bytes/interconnection	Number of interconnections
Fast $ signals	4	Approx. 16000 per bus type
Standard signal	Max. 36 (No. CPUs + 1)* 4)	Min. 1800 per bus type

Table 2-9 Calculating the maximum number of interconnections

NOTE If standard and fast interconnections are combined, an appropriately lower number are obtained.

2.1.6 Significance and uses of the process image

A process image is an instantaneous image of all interface signals from the process at the start of a cyclic task.

Necessity for data consistency

For 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 various tasks must be kept constant during a computation cycle. If this was not the case, interface signal changes while processing a task and run times of the individual function blocks would unpredictably influence the result of a computation cycle.

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

The task administrator (refer to the Chapter "Mode of operation of the task administrator") of the operating system is started with the basic CPU clock cycle T0. This decides which tasks are to be started (T1 and a maximum of additional Tn,

with Tn Ɛ{T2...T5}.

Task processing

Within the task processing, the following components are to be executed:

- Buffer changeover for the tasks to be started (task 1 T1 and if required an additional task Tn)
- **System mode of function blocks in T1 corresponding to the block sequence**
- **System mode of function blocks in Tn corresponding to the block sequence**
- 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 the process image are highlighted; for the other components refer to the Chapter "Description and use of the signal transfer mechanisms".

2.1.6.1 Implementing the process image

System mode

The system mode is used to implement the process image before a task is computed. In the following Fig. 2-6, the sequence in which the function blocks are executed in the system- and standard mode is illustrated in cyclic operation (CPU in the RUN status). In this example, functions blocks 10 and 30 in the system mode are computed within the process image so that the results can be subsequently consistently used in the standard mode.

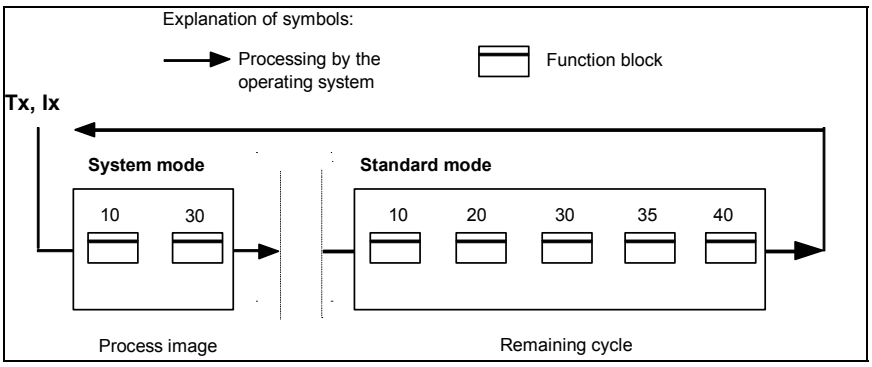


Fig. 2-6 Sequence of the function block computation in the system- and standard modes

The system mode starts immediately after the initiating event (process interrupt or basic clock cycle) in order to create 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 periphery are computed.

2.1.6.2 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 with the blocks in the standard mode of the same cycle.

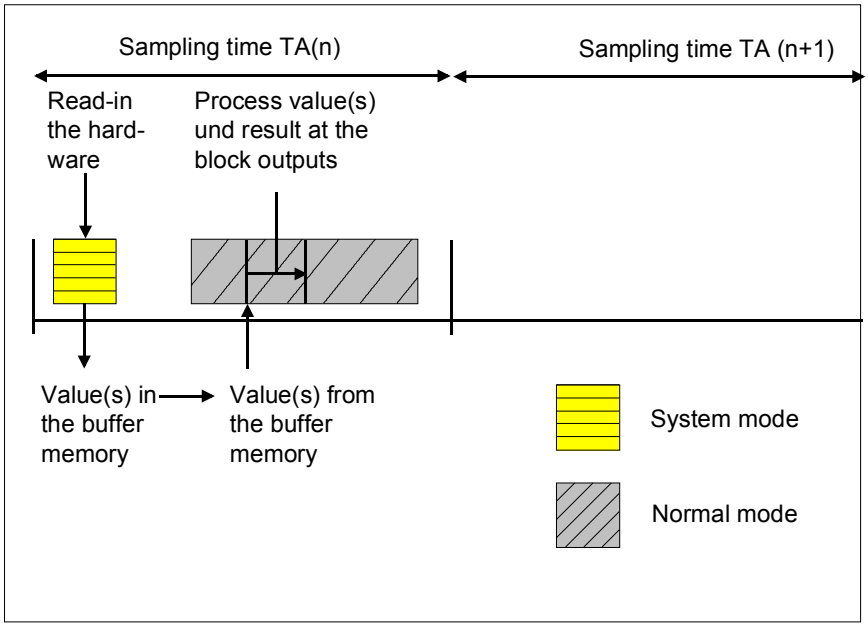


Fig. 2-7 Sequence of the system mode for input blocks

Output blocks with system component

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

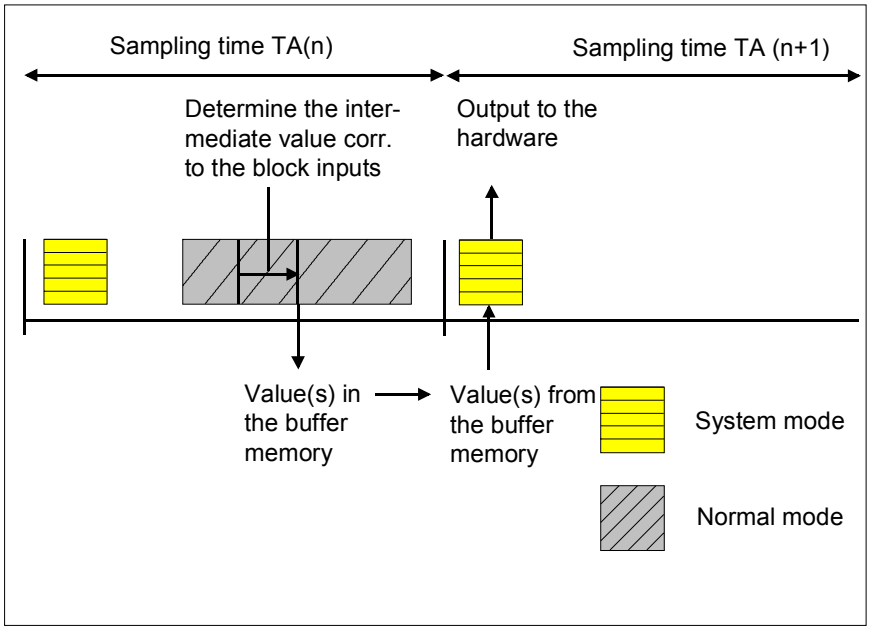


Fig. 2-8 Sequence of the system mode for output blocks

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. For computation in the **standard mode**, the interface signals at the blocks are computed, bypassing the process image within the **standard mode**. For input blocks, the signals are read-in immediately before being computed, and for output blocks, immediately after their computation.

2.1.6.3 Process image for interrupt tasks

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

Mode of operation of an interrupt task

An interrupt task can interrupt a cyclic task running in the standard mode. However it cannot be interrupted by cyclic tasks. Thus, e. g. for longer computation times of an interrupt task, the start of cyclic tasks and therefore output to the hardware can be delayed. This is because, for output blocks with system mode, the signal is only output to the hardware after the next task has been started.

Further 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.

2.1.7 Significance and application of the CPU synchronization

Configuring the CPU synchronization

The CPU synchronization is configured in the HWConfig program section. The directory of the appropriate SIMATIC TDC/SIMADYN D station is opened in the SIMATIC manager and HWConfig is activated by double clicking on the hardware symbol in the righthand 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.**

SIMADYN D synchronizing mechanisms

SIMATIC TDC/SIMADYN D provides the following synchronizing mechanisms:

- Time synchronizing
- Synchronizing its own basic clock cycle to the clock cycle of a master CPU
- Synchronizing its own basic clock cycle to an interrupt task of a master CPU
- Synchronizing its own interrupt task to interrupt tasks of a master CPU
- Synchronizing several stations
- Response when synchronization fails
- Configuring the CPU basic clock cycle synchronization
- Configuring the interrupt task-synchronization

2.1.7.1 Time synchronization

The real-time clocks of all CPUs in a SIMATIC TDC/SIMADYN D station are synchronized to the clock of CPU inserted at slot 1. This prevents the various CPU clocks from drifting apart. This synchronization is automatically realized every 10 s.

2.1.7.2 Synchronizing its 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 subrack and can be received from other CPUs of the station, or by several SIMATIC TDC/SIMADYN D stations, which are coupled using the rack coupling or GDM coupling. For the receiver CPU, an offset can be configured between the basic sampling time and the transmitter basic sampling time. This time offset can also then be changed online with the CPU in the RUN status using the DTS function block type.

2.1.7.3 Synchronizing its own basic clock cycle to an interrupt task of a master CPU

At the start or at the end of an interrupt task of a transmitting CPU, it is possible to initiate an L- or C-bus interrupt. This can be received from one or several other receiver CPUs where it is then used to generate the basic clock cycle.

2.1.7.4 Synchronizing its own interrupt tasks to interrupt tasks of a master CPU

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

2.1.7.5 Synchronizing several SIMATIC TDC/SIMADYN D stations

CS12, CS13 and CS14 modules (master rack coupling) and CS22 (slave rack coupling) (SIMADYN D) or CP52M0, CP52IO, CP52A0 (SIMATIC TDC) and CP53M0 (SIMATIC TDC with SIMADYN D) are available to synchronize the basic sampling time over several stations. In this case, the bus systems of the two stations are connected via coupling modules.

Further information
on synchronization please refer to the "System and communication configuring D7-SYS" Manual.

2.1.7.6 Response when the synchronization fails

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.

2.1.7.7 Configuring the CPU basic clock cycle synchronization

The configuring is set in the dialog window "Basic clock cycle" of HWConfig (refer to the Chapter "Significance and use of CPU synchronization). The synchronization is disabled as default.

Basic clock cycle generated by the CPU itself

If the CPU should generate a basic clock cycle itself, the following settings must be made in the dialog field „Basic clock cycle“ (refer to Fig. Dialog field, basic clock cycle in HWConfig):

- Activate the „Generate“ button with a 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 pre-assigned (default).

Synchronizing the basic clock cycle to a source.

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

- Activate the „Synchronizing“ button with a mouse click.
- Select the required source from a list, e. g.

  L- or C-bus basic clock cycle

  L- or C-bus interrupt (SIMADYN D)

  bus interrupt (SIMATIC TDC)
- Enter an equivalent sampling time of 0.1 to 16 ms.

  Pre-assignment = 1.0 ms (default)
- If required, enter a synchronization delay time of 0.1 ms up to the equivalent sampling time.

  No sampling time is pre-assigned (default value)

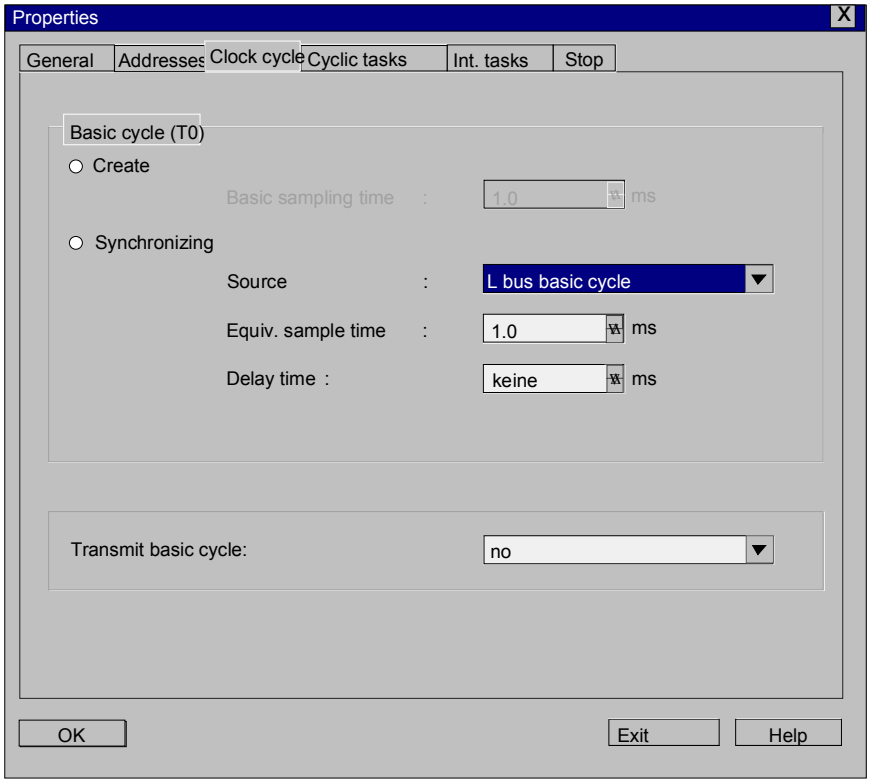


Fig. 2-9 Dialog field, basic clock cycle in HWConfig

2.1.7.8 Configuring the interrupt task synchronization

The setting is made in the dialog window "Interrupt tasks" of the HWConfig (refer to the Chapter "Significance and use of CPU synchronization"). The synchronization is disabled as default, i. e. no process interrupts are defined and a bus interrupt is not transmitted.

Setting the interrupt task synchronization

- The mouse is used to select one of the 8 possible interrupt tasks I1 - I8.
- Select the required source of the defined process interrupt from a list, e. g.

  C bus interrupt or

  CPU counter C1 or C2
- Enter an **equivalent sampling time** from 0.1 to 16 ms.

CPU as interrupt source for the subrack

In the lower window section, select whether the selected CPU is to function as the process interrupt source for the subrack. In this case, one of the defined interrupt tasks I1 - I8 must be selected, and transmitted on the L- and/or C bus. It can be decided as to whether the interrupt task is sent at the start or at the end of the interrupt task processing.

Transmitting at the start of interrupt task processing

It is practical to transmit the interrupt task at the start, if several alarm interrupts must start in synchronism on several CPU modules without any delay. However, the interrupt task on the receiver CPU module may end before the interrupt task on the transmitting CPU module as the transmitting task was inhibited by a higher-priority interrupt.

Transmitting at the end of of interrupt task processing

If transmitted at the end, it is ensured that the task on the receive side is not started before the transmit task has been completed. This second possibility can be used when data is being transferred from a transmit- to a receive task.

2.1.7.9 Example of a synchronization configuration

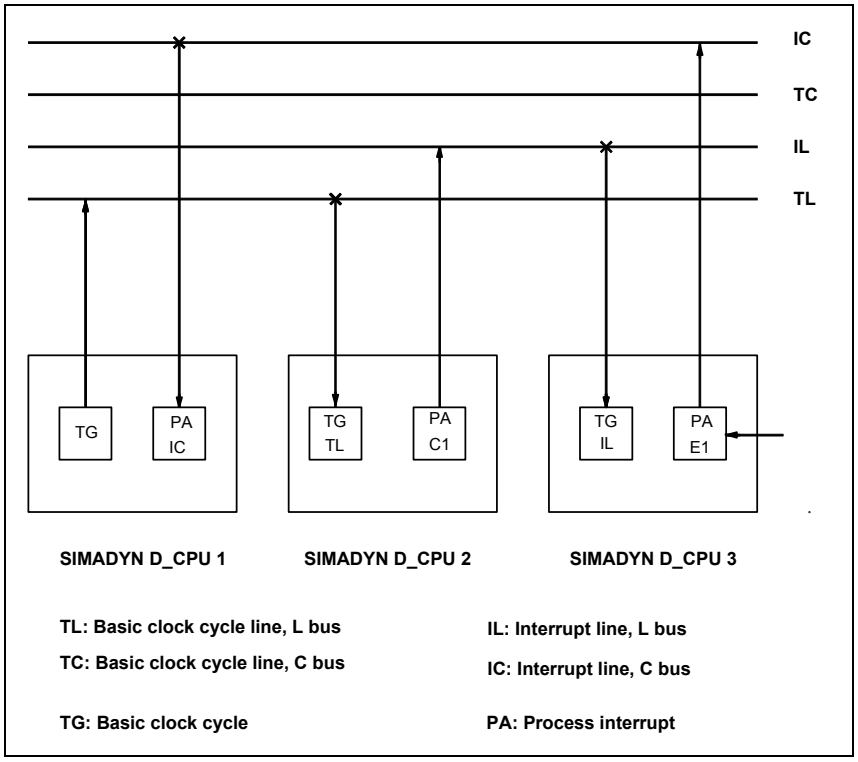


Fig. 2-10 Synchronization configuration

Description

In Fig. 2-10 Synchronization configuration, CPU 1 transmits its basic clock cycle onto the L bus. Further, the C bus interrupt is used as interrupt event by an interrupt-controlled task of the CPU 1.

CPU 2 retrieves its basic clock cycle from the basic clock line of the L bus and switches the interrupt from counter C1 (configuration with function block PAC) to the L bus interrupt line.

CPU 3 retrieves its basic clock cycle from the L bus interrupt line and switches the interrupt, received via the binary input (configuration with function block PAI) to the C bus interrupt line.

2.1.8 Significance of the processor utilization

2.1.8.1 Determining the approximate processor utilization

When compiling, the CFC determines a value for the CPU computation time utilization. A list is accessed, in which the computation time of a block is entered for each function block type. When developing the blocks, these computation times are determined for the "worst case", and are specified in the User documentation, function block library (Edition in Autumn 1997).

For several function blocks, especially for blocks, which access hardware, the worst case situation will generally result in higher time and therefore a typical computation type is used (e. g. for medium bus load levels). Based on these nominal values, for several function block types, the actual computation time can fluctuate significantly.

The computation time, entered in the block catalog, specifies the typical block computation time on a PM5 in µs. However, this value especially for communication blocks, can deviate from the actually required time, depending on the quantity of data to be transferred.

After the charts of a CPU have been compiled using the CFC editor via the menu item **Chart > Compile** the path of a MAP list is specified in an info window or in an error window. The processor utilization, entered in the MAP list, is an approximate value for the reasons mentioned above, which is generally accurate to approximately +/- 10 %.

2.1.8.2 Calculating the precise processor utilization

Function block PSL

The precise CPU utilization can only be determined when the PSL "Permanent System Load" block is configured. The PSL block is configured in any cyclic task in the CPU to be investigated.

It has 5 outputs (Y1..5) which display the actual utilization of the individual tasks in the form of a load factor. The displayed factor should not exceed 1.0 (100%). Values exceeding 1.0 indicate that a CPU is overloaded.

Further, the PSL block has 5 inputs (T1..5) which, for each task, can be used to simulate an additional load in milliseconds (ms). It is then possible to read how such a load effects the utilization of the individual tasks at the outputs. The utilization is determined by measuring the task run times and then dividing this by the actual sampling time. Higher priority tasks occur within the run time of a task which extend the run time and noticeably increase the utilization. Thus, by just adding these values, it isn’t possible to obtain an overall utilization level.

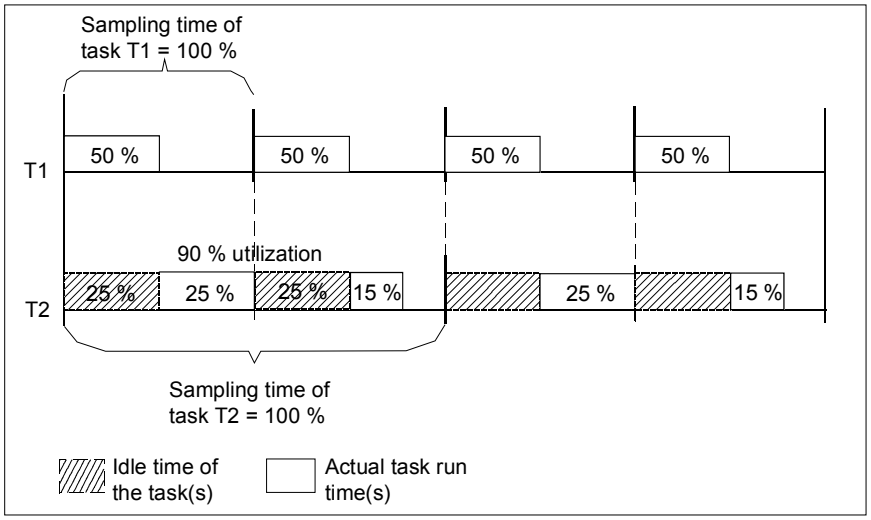


Fig. 2-11 Calculating the run time

2.1.8.3 Mode of operation of the task administrator

The mode of operation of the task administrator is illustrated in this Chapter in Fig. 2-12.

If a task can be completed within a basic sampling time due to a low computation time, then this is illustrated in the 1st cycle.

If a task can no longer be completed within a basic sampling time due to a higher computation time, then it is completed in the following basic cycles. The tasks with short sampling times are completed before tasks with long sampling times, i. e. T1 before T2, before T3 before T4 before .... This distribution is permissible, i. e. without cycle error, as long as the required sampling times are maintained (refer to the 2nd and 3rd cycle).

Cycle errors

If the computation time loading becomes higher, for the task with the longest sampling time, at same stage a cycle error will occur. This means that the sum of the function blocks cannot be computed completely within the configured sampling time.

NOTE

If a specific number of cycle errors is exceeded, an "E" error ID is set, and is displayed in the 7-segment display on the front panel 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, especially by communication interrupts. These interrupts ensure that, for example, 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 can occur independently of the configured cycle time of the appropriate communication blocks at almost any instant in time. As result of this, and the unpredictable occurrence of interrupt tasks, if the process utilization is extremely high, each cyclic task can generate one or several cycle errors due to task back-up.

This can be especially noticed, if

- the utilization by the task with the lowest sampling time is extremely high, and
- the functions computed in this task are extremely sensitive to sporadic sampling cycle failures, (e. g. closed-loop position controls).

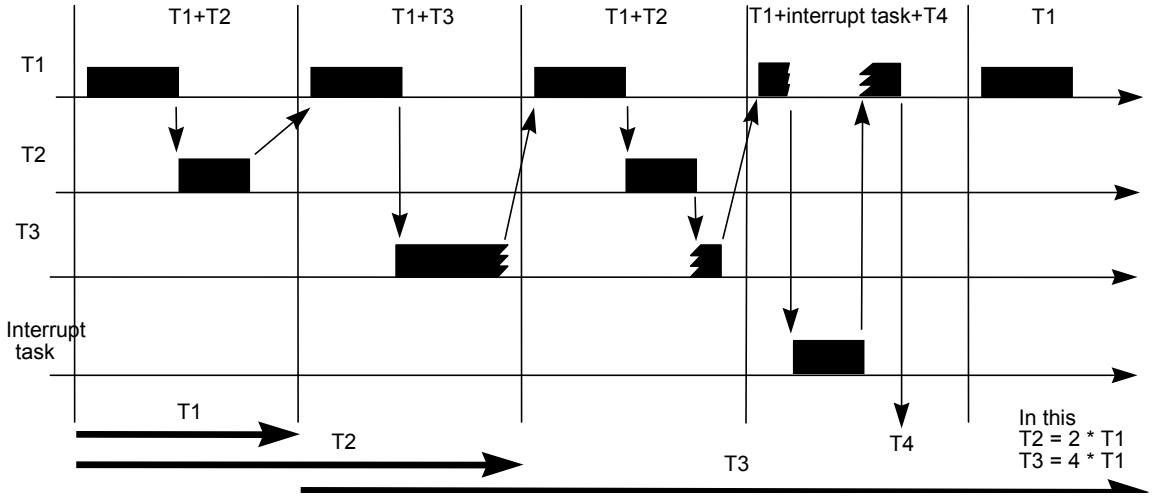


Fig. 2-12 Sequence of a configured task

2.1.8.4 Eliminating cycle errors

The modular SIMATIC TDC/SIMADYN D provides the following possibilities of eliminating cycle errors:

- Increasing the configured basic sampling time
- Shifting configured blocks from fast to slow tasks.
- Using several or higher-performance CPUs or several SIMATIC TDC/SIMADYN D stations
- Reducing the number of blocks or changing the block types
- Checking the necessity to have communication interfaces on this CPU
- Checking the necessity for interrupt function packages on this CPU

NOTE On a case for case basis it should be checked the most cost-effective way to achieve the desired result.

2.1.9 Technical data of the operating system

2.1.9.1 Features

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

Number of CPU modules

A maximum of 8 (SIMADYN D) or 20 (SIMATIC TDC) CPU modules can be inserted in a subrack. A CPU module requires 1 slot. Slots which are not occupied by CPU modules can have peripheral modules.

Number of function diagrams

The maximum number of function diagrams is dependent on the particular software, but is approximately 65536.

Cyclic tasks

System diagram	Available automatically
Basic sampling time T0 can be configured	From 0.1 [ms] to 16 [ms] in steps of 0.1 [ms]
Number of configurable cyclic tasks	5
From the basic sampling time	T0
To	T0 * (2 ** 15)
Configurable from	T0 to 32768 * T0 e. g. of 1 [ms] to 32768 [s]

Table 2-10 Technical data of the cyclic task

Interrupt tasks

Number of configurable interrupt tasks		8
Number of available interrupt sources, total		54 (SIMADYN D) or 19 (SIMATIC TDC)
Of which		
	Software interrupts	8
	CPU timer interrupts	2
	Interrupts for binary inputs	4
	Bus interrupts (L/C)	2 (SIMADYN D) or 3 (SIMATIC TDC)
	LE bus interrupt	4 (only SIMADYN D)
	LE bus interrupt, extended	32 (only SIMADYN D)
	Only T400 ISL, ISR	2

Table 2-11 Technical data of the interrupt task

Computation times of the operating system

The run times of the operating system are specified in the following, based on the PM5 CPU module. For PM6 CPU modules, the computation time is shortened to approximately one third of the specified times.

The signals, which are transferred along the L- and/or C bus represent an almost consistent system load, as the bus is always clocked at 8 MHz.

The minimum time is shown in the following table which is required to process each cycle of a task (refer above for the basis for the calculations!):

Time to start	40 µs
Time to end	40 µs
Additional component for a local buffer system	20 µs
C-bus buffer system	20 µs
L-bus buffer system	20 µs

Table 2-12 Computation times of the operating system

Memory requirement of the operating system

The code and data of the operating system are copied from the memory module into the CPU RAM on the CPU module and the data is „unzipped“. Memory requirements are as follows:

- CPU-RAM area: 400 Kbyte
- Memory module area: 200 Kbyte („zipped“)

On the communication buffer modules, the operating system after the start uses 1 Kbyte of the C-bus- and the L-bus buffer memory as data area to administer operating system lists. This is supplemented by the appropriate memory requirement, depending on the configured software, for the buffer system and additional components, e. g. communications.

Operating system version

The operating system is identified by a version ID in the form „yymmddVxyz“. The significance of the individual letters is

- yy: Year, mm: Month, dd: Day
- "V": Version
- xzy: Version number

e. g. for version 5.0 as "961201V500".

2.1.9.2 The basic operating system functions

Operating system components

The operating system is comprised of the following components:

- Task administrator for cyclic- and interrupt controlled processing
- Hardware and software initialization
- Memory administration (buffer administration)
- Operating system data and lists
- Interface to the central AMC 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, whereby these represent the most important interfaces to the environment.

Operating system functions	Initiated by
Initialization	RESET
Cyclic processing	Sampling time timer
Interrupt-controlled processing	Process interrupts
Process image	
Exception handling and diagnostics	System interrupts
Communications, I/O	Input/output interrupts
Service	
User program	
Utility programs	

Table 2-13 Basic operating system functions

Initialization

Initialization is initiated by powering-up the power supply or depressing the RESET button to output a reset pulse. The initialization conditions/prepares the hardware and software so that the system can go into the standard operating mode (RUN status).

Cyclic processing (RUN operating status)

The task administrator ensures that the functions, assigned to the various tasks are cyclically processed. The cyclic tasks are in a ratio to the power of 2 to each other

T(i) = T(0) * (2 ** j) with T(0) as basic sampling time,
j defines the sampling time value with 0 <= j <= 15
i numbers the sampling times with 1 <= i <= 5.

Example:

For a basic sampling time of 1 ms, the sampling times can be 1 ms, 2 ms, 8 ms, 32 ms and 128 ms. The basic sampling time is defined for each CPU module during configuring, using the HWConfig program section of the SIMATIC Manager. The sampling times of the tasks running on the particular CPU module are also configured at this time.

In order to prevent bottlenecks, the tasks are started, phase-shifted with the basic clock cycle, so that with the basic clock cycle, the start of a second, lower-priority task is flagged. As result of the discrete distribution of the sampling times, based on a ratio to the power of 2, also low-priority tasks are completely taken into account. This means that it no longer occurs as a low-priority sampling time on the basic clock cycle. (refer to the Chapter "Processor utilization"). The priorities of the various tasks decreases with increasing sampling time.

The task administrator is started with the clock cycle of the basic sampling time of the sampling time timer. This determines the second task, task Tn to be started in addition to T1 (Tn from {T2...T5}). 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 determined task is started. The status of the interrupted task is written

into a task-specific data area, which allows the task to be further processed as soon as a higher-priority task is no longer present (refer to Fig. Calculating the run time).

The time component required by the operating system itself is not taken into account in this description. If the diagram was to be precise, then the actual starting instant of the task would be shifted by these amounts.

Interrupt-controlled processing

In addition to cyclic processing, the operating system also administers tasks, which are started by non-cyclic interrupts, especially process interrupts. Interrupt sources could be:

- software interrupts
- CPU-timer interrupts
- L/C-bus interrupts
- LE-bus interrupts

The priority of the interrupt tasks is defined by the data configured in HWConfig (I1 > I2...> I8). The programming engineer programming the user program configures, using HWConfig, the interrupt sources which he or she requires for his or her application, and their processing in the interrupt-controlled tasks.

Process image (system mode)

Before a task is processed, it is first investigated whether an associated process image must be updated. If yes, this is realized before the task is started, by calling-up the system mode of the function blocks (refer to the Chapter "Significance and use of the process image"). The update is referred to:

- binary inputs/outputs, for example, the status images for controller enable signals and the position of limit switches.
- analog inputs/outputs, for example, values for temperature, speed, etc.

NOTE

The system mode is started for both tasks to be started before standard mode processing (refer to the Chapter "Significance and use of the process image").

Error differentiation

SIMATIC TDC/SIMADYN D differentiates between errors, which occur during initialization and those which occur during standard operation.

Errors from the initialization (INIT operating status) result that the system is not released for start (transition into the RUN operating status).

For errors in standard operation (RUN status), a differentiation should be made whether processing is to be continued or terminated.

The system informs the user about its status, especially about the error statuses, using the 7-segment display on the CPU module.

When an error situation occurs, detailed information is deposited in the error data fields of the operating system. These error data fields permit a precise error analysis to be made.

This data can be read-out and changed using the service utility.

The significance of the error signals and information can be taken from the online help "D7-SYS, Help on events".

Communications Communications handles all of the input/output data transfer between the hardware as well as the associated software components and the user interfaces. The interfaces and their parameterization are configured in the user program using CFC.

Service utility The service utility is the central interface of the CPU modules. It is an instrument for start-up, diagnostics and troubleshooting.

As the processing time of the service utility is undefined, the task associated with it as well as the tasks with lower priority can be blocked. This has been implemented, so that service is allocated a maximum processing time within its cycle (maximum of one basic clock cycle T0).

The service units form the user interface via which the communications software is controlled.

User program The user program is used to implement the technological tasks on the target hardware. It is generated at the programmer in the CFC programming language, using the available utility programs such as HWConfig, CFC editor, CFC compiler, linker/locator and the memory module driver.

The CFC source code of the user program is converted into data structures using the CFC compiler, and loaded on the target hardware where it is processed by the operating system.

Utility programs Utility programs are basic system functions for the operating system. These include watchdog functions, functions to handle the CPU display, special test- and interrupt routines to handle system errors.

2.1.9.3 The service utility

The service utility provides a pool of information functions so that the user has access to system information on the processor. The service utility is designed as support resource for start-up and testing.

Start-up area Configured data (setpoints/actual values) are displayed and/or changed here and the software optimized (e. g. interconnection changes, controller times changed etc.).

Testing Causes of plant/system faults (crashes, run-up problems) and faults, which are caused in the CPU module itself, are identified here.

All activities of the service utility are controlled via tasks, which are received via "its" data interface (corresponding to the parameterization of the service function block I/O).

All devices which can process the task- and response language of the utility can be used as handling devices for the service utility. In the SIMATIC TDC/SIMADYN D world, these are the programs (tools) CFC in the test mode and service IBS (service start-up).

NOTE The user can also use his own tools. They must be compatible with the interface definitions of the service utility. The interface specification can be sourced from ASI 1 R.

The service utility is made available with the "SER" function block. This function block ensures that none of the messages/data get lost.

Task processing

The service utility differentiates between cyclic and non-cyclic tasks. A non-cyclic task is completed when its response telegram has been sent.

A cyclic task remains active until it is explicitly terminated, either by being aborted via a reset or as result of a new task. A task comprises of at least one response telegram.

NOTE The service utility can always only process one task. The next task is only processed if the previous task was responded to.

System loading, response times

The actual service utility processing is realized in a 32 ms sampling time (the next sampling time below 35 ms is selected; the sampling times specified at the SER blocks are not significant for processing). In the cyclic task used, the service blocks are provided a certain computation time, which may use as maximum the basic clock cycle T0. The ratio of the basic clock cycle T0 to the used task defines the CPU performance available and therefore the system loading.

Example 1:

Basic clock cycle T0 = 1 ms; selected sampling time = 32 ms. Every 32 ms, 1 ms is reserved for the service utility. Thus, the system load is calculated of

1ms / 32 ms = 0.03125 = 3.125%

Example 2:

Basic clock cycle T0 = 2ms; selected sampling time = 16 ms. Every 16 ms, 2 ms are reserved for the service utility. Thus, a system load is obtained from

2ms / 16 ms = 0.125 = 12.5%

This available computation time is used by all service blocks to the same extent, i. e. as long as the time is sufficient, if possible, all of the SER blocks are processed once. An SER block processes a maximum of one task per clock cycle. For cyclic tasks, for each clock a maximum of one response telegram is received. The advantage of this mode of operation is that for cyclic tasks, equidistant-timed responses are obtained.

If the reserved computation time is not fully utilized, because, for example, there is no task to be processed, then this time is made avasilable to the system.

For multiple configuring with simultaneous access to system resources which are only available once (e. g. change memory of the memory module), resources are assigned to the first component which makes the request. All others are rejected and output at the latest after 1 second, an error message ("resource occupied") via the data interface.

Behavior under fault conditions

In a fault condition (exception), i. e. for initialization errors or online faults, the system goes into the stop mode. Thus, there are special conditions for the service utility. It is then no longer computed in a cyclic task, but runs continuously, started from an exception administrator. Under fault conditions, the service utility cannot be connected to the configured user. In order to still permit system diagnostics, the CPU's own diagnostics interface is connected. The DUST1 protocol runs here (refer to the Chapter "Operating statuses of a CPU module").

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 available for diagnostic purposes.

NOTE Before investigating a fatal system error, the INIT_ERR and SYS_ERR system error fields should first be investigated. If errors are entered there (especially hardware (monitoring) errors), then this could be the cause of a fatal system error.

A SAVE area is set-up in the upper area, in the local RAM of each CPU module. This area is not erased at re-initialization, if the status of the RAM copy is appropriate. An error buffer is set-up in this SAVE area, which includes the error protocol (error report) consisting of several messages.

The error buffer consists of an administration part and a ring buffer, in which the error messages are saved. The ring buffer is implemented as buffer which can be overwritten, i. e. if the buffer is full with error messages, then the new messages overwrite the oldest messages.

There are 2 different types of error messages. A long message is output in the case of a non-maskable interrupt NMI. A short message for a power-OFF.

The service communications utility is available, (even if it has not been configured) to troubleshoot fatal system errors. It can be accessed via the local diagnostics interface, after pressing the acknowledge button. Using the service utility, the error causes can be output in plain text.
What is especially important is the error cause, specified under an ID code and supplementary ID. If a function block is being calculated at the instant that the system error occurs, then this is output. In addition, the results of the last bus accesses are displayed; these are important, if a bus access is the error cause. Further, all of the process registers are displayed for the system specialists to allow them to make a precise error analyses.

NMI handling

When a non-maskable interrupt occurs, this is considered as fatal error and causes initialization or normal operation to be interrupted. All of the modules, inserted in the subracks, are no longer processed.

A large flashing H is displayed on the CPU module display of the faulted module, which caused this fatal error. A large H is displayed as steady display on the other CPU modules which received an NMI as result of the faulted module. The debug monitor can be activated by pressing the acknowledge button or setting the status value.

The symbols output on the 7-segment display have the following significance:

Steady H : CPU module was shutdown by another module.

Flashing H : Fatal error on this CPU module (error cause).

Example for an error protocol (error report) for fatal system errors

```
Information on the last crash:
Time:                 01.01.93  04:16:24.9294 h        Crash instant
ID:                   #5 CPU
Supplementary ID:     28 (unaligned instruction fetch) } Crash cause
EPC:                  0x04C4F19A                       Crash address and
Return jump address:  0x801201f8                       address of the function call

Running task:         T2
Started levels:       T2(NRM),T5(NRM),BACKGROUND       The running and started tasks
Last processed FB:    FP-KRUMMS.AY0815 (Typ:  ADD8F)   at the crash instant as well as
                      (ALE: 0x80107D84  CODE:          the processed function on the
                                        0x801201E0)    data- and code areas

Last L-bus access:                                     Data on the system bus accesses.
- Access type:        q_read_2byte                     This data is especially interesting,
- BUS address:        0xB0F25874                       if an erroneous bus access is
- Retries:            0                                specified as the cause of the crash
Last C-bus access:                                     specified (NMI) and supplementary
- Access type:        q_read_2byte                     ID.
- BUS address:        0xB4F400B4
- Retries:            0                                Register dump of all process
                                                       registers
The processor status at the crash instant:             Especially important:
EPC     : 0x04C4F19A    BadVAddr: 0x04C4F19A           EPC (crash address, as above) and
Status  : 0xF000FC14    mdlo    : 0x04C4F19A           BadVAddr (bad virtual address,
fpc_csr : 0x00000F04                                   address which was erroneously
                                                       accessed(mainly for ID TLB and
                                                       CPU))
CAUTION: The value of a0, a1 (and possibly a2) is not
valid!

r00/0 : 0x00000000  r01/at: 0x00000000  r02/v0: 0x00000001  r03/v1: 0xB8803000
r04/a0: 0x80064FC8  r05/a1: 0x80064F44  r06/a2: 0x0000000A  r07/a3: 0x00000000
r08/t0: 0x80064FC4  r09/t1: 0x80065048  r10/t2: 0x19999999  r11/t3: 0x00000000
r12/t4: 0x04C4F19A  r13/t5: 0x8007FE90  r14/t6: 0x00000000  r15/t7: 0x00000000
r16/s0: 0x800650CC  r17/s1: 0x8006511C  r18/s2: 0x00000000  r19/s3: 0x8006548C
r20/s4: 0x8006548C  r21/s5: 0x00000020  r22/s6: 0x800812E0  r23/s7: 0x80400000
r24/t8: 0xFFFFFFFF  r25/t9: 0x8007FE90  r26/k0: 0x00000210  r27/k1: 0x04C4F19A
r28/gp: 0x80088BA0  r29/sp: 0x80064EA8  r30/s8: 0x04C4F19A  r31/ra: 0x801201f8

d00:       +Denorm  d02: +6.400000e+01  d04:       +Denorm  d06: +9.999000e-01
d08: +4.687500e+01  d10:       +Denorm  d12:       +Denorm  d14:+2.660285e+154
d16: +3.000001e+03  d18: +4.687500e+01  d20:          QNaN  d22:+8.329648e+298
d24: -1.818767e-12  d26:-1.227518e+306  d28:-3.691391e+249  d30:-2.374690e-237

f00: +4.787490e+01  f02: +0.000000e+00  f04: +4.687500e+01  f06: +4.764729e+05
f08: +0.000000e+00  f10:       +Denorm  f12: +4.787490e+01  f14: -1.960343e+37
f16: +1.024000e+03  f18: +0.000000e+00  f20: -1.693935e+38  f22:          QNaN
f24:          QNaN  f26:          QNaN  f28: -6.835168e-27  f30: -4.550802e-04

---------------------------- End of the diagnostics -----------------------------
```

Causes of fatal error

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

Supplementary ID code (precise description)

NMI a non-maskable interrupt
- second bus clear for task-controlled access
- bus clear for direct access
- timeout during L/C-bus arbitration/assignment (module missing/defective, daisy chain missing)
- ready internal from L/C bus (error on another CPU module)
- ready internal from the local expansion bus (LE bus)
- system bus controller overrun
- timeout when accessing the local periphery
- spurious interrupt (an interrupt source cannot be identified)
- direct access to the L/C bus (bypassing the driver functions)

CPU exceptional status of the CPU
- internal error
- reserved Instruction
- unknown Syscall
- unaligned instruction fetch (jump to address which cannot be divided by four)
- user access to kernel space
- unaligned load/store to coprozessor 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 running in endless loop

FPU FPU exception status
- fpu fault at non-fpu instruction
- illegal fpu sub opcode
- operation on NaNs
- add/sub/division of infinities
- mul of infinity and 0

TLB exception status of the TLB
- TLB modified exception
- TLB read/write miss (access to illegal address)
- UTLB miss (access to illegal address)

TIME basic cycle time failure

OFF power down
- power down/reset in the normal mode
- power down/reset in the stop mode (after another exception)

2.2.2 Background processing

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

As background task, the following functions are simultaneously available:

- the online test mode and
- 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, then only the service communications utility is activated.

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

2.2.2.1 Online test mode

In the online test mode, for example, a battery test, a memory module checksum test etc. are executed. The memory module checksum routine determines the memory module checksum and compares it with the checksum calculated by the programmer and that saved in the memory module. If a memory module checksum error is identified in the online test mode, the user can remove the error by repeatedly generating the memory module. For battery test errors, he can replace the battery.

2.3 System chart @SIMD

Overview

The system chart @SIMD (Part A and B) is a CFC chart made available as standard to the user. They permit standard diagnostics of the hardware and system software.

Program structure

The system chart is configured, structured in the following parts

- Acknowledge — Acknowledge the error display
- Evaluate components — Determine the components which signaled an error
- Display — Output the identified error

System chart @SIMD	
	Function block names
Acknowledge	
Pushbutton	ACK Acknowledge
Service intervention	ACK
Evaluate components	
First error field	FER First error
Communications error field	CER Communication error
Task administrator error field	TER Task management error
Hardware failure Monitoring error	HER HW error
User error field	UER User error
Evaluate errors	DER Display error
Display	
Output, 7-segment display	DST Display status
Output, diagnostics LED	DST
Output, status word SIMS, status bit SIMD	SIMS, SIMD

Table 2-14 Detailed information on system chart @SIMD

Description

The operating system monitors the hardware and 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 allow the user access to these flags. An output is displayed on the 7-segment display of the CPU module if a flag of a component was set.

If several messages are generated for the 7-segment display, the highest-priority message is output.

Error name	Error display	Priority
Communications error	C	High
Task administrator error	E	
Hardware failure, monitoring error	b	
User-generated error ID	A	
No error present	CPU number	Low

Table 2-15 Error priorities for the message display

Resetting the flags

The flags of the displayed error and the next priority error code is displayed when the acknowledge button on the CPU module is depressed, or an acknowledge signal is issued via a service unit. If there are no errors present, the CPU number is displayed on the 7-segment display as the lowest priority message. In order to identify that the displayed error message was the first to occur, it is displayed flashing.

Mode of operation

The sequence diagram illustrates the global program sequence of the system charts. It consists of the three functional components

- identify acknowledge signal
- evaluate components, and
- display.

Identify acknowledge

The acknowledge signal is a pulse which is derived from the pushbutton status read-in from the ASI function block or as result of a service intervention at connection 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 connection from 0 to 1.

Evaluating components

The components are evaluated using the function blocks SYF1 and SYF4. The appropriate numbers of the errors fields are documented in the function block description (refer to the reference manual, SIMADYN D function block library). An error field can only be acknowledged if an error was identified for the particular component and this was displayed.

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 in the first error field is displayed flashing on the 7-segment display.

All of the components are evaluated according to their priority one after the other. The communications error field cannot be acknowledged, as a software change is required in order to remove this error. When the system runs-up, the CPU could be subject to a higher loading. Task administration errors are automatically acknowledged during the system run-up using a counting logic function.

Control using priority logic

A priority logic circuit ensures that only the highest priority component is displayed. The lowest-priority component supplies a bit signal, which changes-over the display from a CPU number display to an error display (UER070.Q). If the highest priority error component is additionally entered in the first error field, the error display is output flashing.

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

NOTE

If a displayed error is acknowledged, the error source is still present. Before an error can be removed, the error cause must be determined and removed.

Display

When there are no errors, the processor number is displayed on the 7-segment display. If a component signals an error, then the appropriate error code is output.

The status display on a T400 is realized via a diagnostics LED. The flashing clock cycle is increased if the error is a first error.

The status display on a FM 458 is realized via fontside LEDs (refer to User Manual "Application Module FM 458").

Sequence diagram

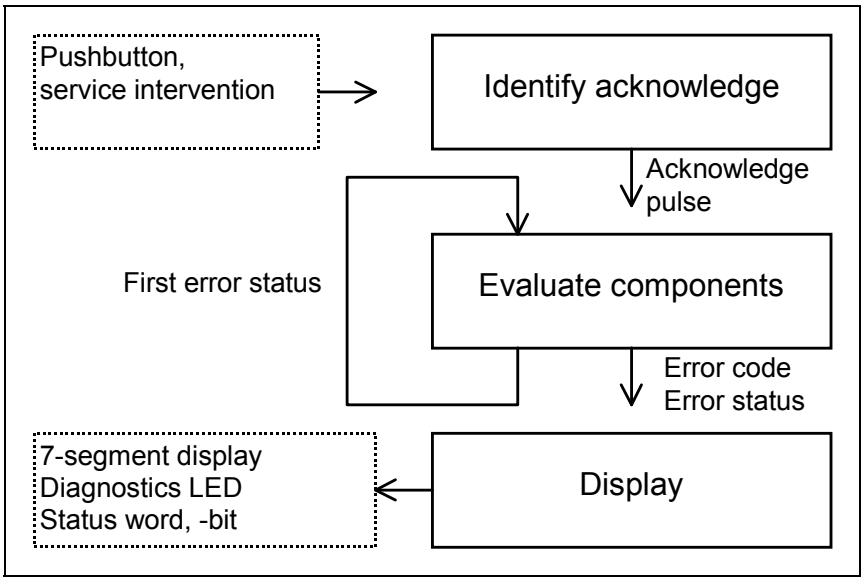


Fig. 2-13 Sequence diagram

Interfaces

The acknowledge button of the CPU module or the possibility of acknowledging via the service interface is provided as external input of the system chart. The 7-segment display of the CPU module or the diagnostics LED (T400) are available as external outputs for the user display.

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

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

Bit	Bit assignment
Bit1	Unused
Bit2	Unused
Bit3	Unused
Bit4	Task administrator error
Bit5	Unused
Bit6	Hardware failure
Bit7	Communications error
Bit8	Unused
Bit9	Unused
Bit10	Unused
Bit11	User-generated error ID
Bit12	Unused
Bit13	Unused
Bit14	Unused
Bit15	Unused
Bit16	Unused

Table 2-16 Bit assignment of the function block connection SIMS.QS

3 Communications configuring

Overview

3.1	Introduction	3-2
3.2	Couplings on the subrack	3-19
3.3	Subrack coupling CP52M0	3-22
3.4	Subrack coupling CP53M0	3-28
3.5	TCP/IP coupling (CP51M1)	3-36
3.6	TCP/IP coupling (CP5100)	3-45
3.7	PROFIBUS DP coupling (CP50M1)	3-53
3.8	PROFIBUS DP coupling (CP50M0)	3-65
3.9	MPI coupling	3-101
3.10	Table function	3-102
3.11	Communications utility, message system	3-138
3.12	Communications utility process data	3-153
3.13	Communications utility service	3-171
3.14	Communications utility time of day synchronization	3-174
3.15	Communications with SIMATIC Operator Panels	3-176
3.16	WinCC connection to SIMATIC TDC via the standard channel (SIMATIC S7 Protocol Suite.CHN)	3-184
3.17	Communications with WinCC (TCP/IP)	3-216
3.18	Communications service Trace	3-226

3.1 Introduction

3.1.1 Basic information on communications

General information

Communications permit information and data to be transferred to other systems and devices.

To establish communications, the following are required:

- a communications utility must be configured together with a link
- a communications interface must be available

Communications utility

The communications utility defines the information contents (e. g. process data) during communications.

Coupling

The coupling defines the hardware (e. g. CP50M0) and the data transfer protocol (e. g. PROFIBUS DP) for communications.

Couplings and communication interfaces

The particular application and communication capabilities of the partner define the communications interface and the data coupling.

3.1.1.1 Overview of the various data couplings

General

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

CPU-local coupling

Used for the communications partner	• CPU-internal to test transmitters/receivers
Hardware required	• CPU
Communications utility	• Process data
Central coupling block	• @LOCAL
Features	• SIMATIC TDC-internal memory coupling

Table 3-1 CPU-local coupling

Direct CPU-CPU coupling

Used for communication partner	• CPU-CPU communication for higher data quantities as an alternative to $ signals
Hardware required	• CPU
Communication service	• Process data
Central coupling block	• @LOCAL
Features	• SIMATIC TDC internal memory coupling

Tabelle 3-2 CPU-CPU coupling

Communications buffer coupling

Used for the communications partner	• CPU-CPU communications for higher data quantities as an alternative to $ signals
Hardware required	• Communications buffer module (CP50M0 / CP50M1 / CP51M1 or CP53M0)
Communications utility	• Process data
Central coupling block	• @GLOB
Features	• SIMATIC TDC-internal memory coupling

Table 3-3 Communications buffer coupling

Subrack coupling SIMATIC TDC

Used for the communications partner	• SIMATIC TDC
Hardware required	Communication modules for the master interface: • CP52M0 (GDM-Memory) • CP52I0 (GDM-Interface)
Hardware required	Communications module for the slave interface: • CP52A0
Communication utility	• Process data, message system, trace
Central coupling block	• @SRACK
Features	• Fiber-optic cable • Parallel coupling of up to 44 SIMATIC TDC subracks • All subracks can be synchronized • Uniform system clock possible (unified) • Fast • The maximum distance between 2 subracks is 200 m • The subrack can be disabled (disconnected) at any time

Table 3-4 Subrack coupling SIMATIC TDC

Subrack coupling SIMATIC TDC with SIMADYN D

Used for the communications partner	SIMATIC TDC with SIMADYN D
Hardware required	Communication module SIMATIC TDC: CP53M0
Hardware required	Communication module für SIMADYN D: CS12/CS13/CS14/CS22
Communication utility	Process data
Central coupling 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 subracks All subracks can be synchronized Uniform system clock possible Fast The maximum distance between 2 subracks is 200 m The subrack can be disabled (disconnected) at any time

Tabelle 3-5 Subrack coupling SIMATIC TDC with SIMADYN D

TCP/IP coupling

Used for the communications partner	• SIMATIC TDC • SIMATIC S5/S7 • Third-party systems • WinCC
Hardware required	• CP5100/ CP51M1 communications module
Communication utility	• Process data and message system • Service • S7 communikation
Central coupling block	• @TCPIP
Features	• Standardized bus according to Ethernet (IEEE 802.3) • Baud rate: 10 or 100 Mbaud (autosensing)

Table 3-6 TCPIP/IP coupling

PROFIBUS DP

Used for the communications partner	• SIMATIC S5/S7 • SIMOVERT/SIMOREG drive converters • ET200 • SIMATIC TDC • Certified third-party equipment/devices
Hardware required	• CS7 communications module with CP50M1/ CP50M0 communications module
Communications utility	• Process data • Parameter processing
Central coupling block	• @PRODP
Features	• Standardized multi-master bus for communications between SIMATIC TDC and a maximum of 123 communication partners • Master slave principle (CP50M1/ CP50M0 is master and/or slave) • PROFIBUS standard according to EN 50170 • Fast • Max. 12 Mbaud • Maximum net data length, 244 bytes • Bus is parameterized using the COM PROFIBUS software(only CP50M0)

Table 3-7 PROFIBUS DP coupling

MPI

Used for the communications partner	• CFC • WinCC • SIMATIC-OPs
Hardware required	• CP50M1/CP50M0 communications module
Communications utility	• Service • S7 communications
Central coupling block	• @MPI
Features	• Multi-master bus with a maximum of 126 nodes • 187,5 kbaud / 1,5 Mbaud • Standard for SIMATIC S7

Table 3-8 MPI coupling

3.1.2 Overview of the communication utilities

General

Various data can be transferred via the communication interfaces, for example, process data and messages.

The communication utilities define which information/data is to be transferred. The communication utilities are defined by configuring the communication modules.

Communication utilities

Communications utility	Description	Communication blocks to be configured
Message system	Establishing alarm- and fault systems	Special message blocks: @MSC, MER..., MSI...
Process data	Transferring process data (setpoints and actual values)	Send- 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 of the CPUs used (e. g. in order to compare messages with a time stamp).	Special function blocks: RTC...
Data trace	Trace process quantities	@TCP, TR...
S7 communications	Operator handling and visualization of CPU program / CFC	Communication function block: S7OS

Table 3-9 Overview of the communication utilities

3.1.3 Communication block I/Os

3.1.3.1 Initialization input CTS

Communication blocks which access a data interface have a CTS input.

Data at the initialization input

The following are specified at the CTS input:

1. The configured name for the communications module

   Syntax for module names:
   - the name is 1 - 6 characters long
   - 1st character: A - Z
   - 2nd - 6th characters: A - Z, 0 - 9, _

2. Connector of the data interface if the data interface is on a CP50M0, CP50M1 or CP51M1 communications module

   Syntax for the connector designation:
   - enter "." after the module name
   - the name after "." is 3 characters long
   - "X01", "X02" or "X03"

Example of data entry at CTS

Configuring example of a subrack:

Slot	Module	Configured module name in HWConfig	Possible data entry at the CTS input
S01	CPU550	"D01P01"	"D01P01"
S03	CP50M1	"KOPPEL"	"KOPPEL"
S04	CP50M1	"KOMM1"	"KOMM1.X01" "KOMM1.X02"
S06	CP51M1	"TCPIP"	"TCPIP.X01"

Table 3-10 Configuring example of a subrack

3.1.3.2 Address connections AT, AR and US

General

Communication blocks, which can access a data interface, have an address connection.

Address connection types

Depending on the particular block type, a differentiation is made between three address connection types:

- AT connection: Available when transmitting
- AR connection: Available when receiving
- US connection: Available for function blocks, which are processing a send- and a receive channel

Possible address connection data

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

- "Channel name"
- "Channel name. Address stage 1"
- "Channel name. Address stage 1. Address stage 2"

Channel name

- The channel name addresses a channel at a data interface.
- Transmitter and receiver, which access a data interface with the same channel name, communicate with one another.
- The channel name consists of a maximum of 8 ASCII characters, excluding "Point" and "@".

NOTE

It is not checked as to whether a channel name is configured a multiple number of times. The configuring engineer must uniquely assign the channel names at a data interface for each transmitter/receiver at the AT, AR or US connections. If this condition is not fulfilled, then

- transmitter/receiver may be used a multiple number of times, but uncoordinated.
- the transmitter/receiver could log-off with a communications error.

Exceptions:

- Several transmitters are permitted for the "Select" data transfer mode.
- Several receivers are permitted for the "Multiple" data transfer mode.

Address stages

- There is address stage 1 and address stage 2.
- Several couplings, for example, PROFIBUS and Industrial Ethernet require the address stage to be specified for data transfer. For subrack couplings, for example, address stages are not specified.
- Address stage 1 is a maximum of 14 characters long, address stage 2, maximum 20 characters.
- Significance and contents of the address stages are described for the particular coupling.

3.1.3.3 Data transfer mode, MOD input

Overview

There are five various data transfer modi for the various communication requirements:

- handshake
- refresh
- select

- multiple
- image

Selecting the data transfer mode

The data transfer mode is specified at the MOD connection of the appropriate transmitter or receiver.

"Handshake" data transfer mode

The "Handshake" data transfer mode is used,

- if information loss may not occur due to data being overwritten, and
- if there is precisely one receiver for each transmitter.

"Handshake" defines a sequential channel processing. The transmitter first deposits a new data set in the channel after the receiver has acknowledged that it has received the first data set. A net data buffer is provided for data transfer.

The transmitter inputs the net data into the channel in an operating cycle and the receiver reads them out in an operating cycle.

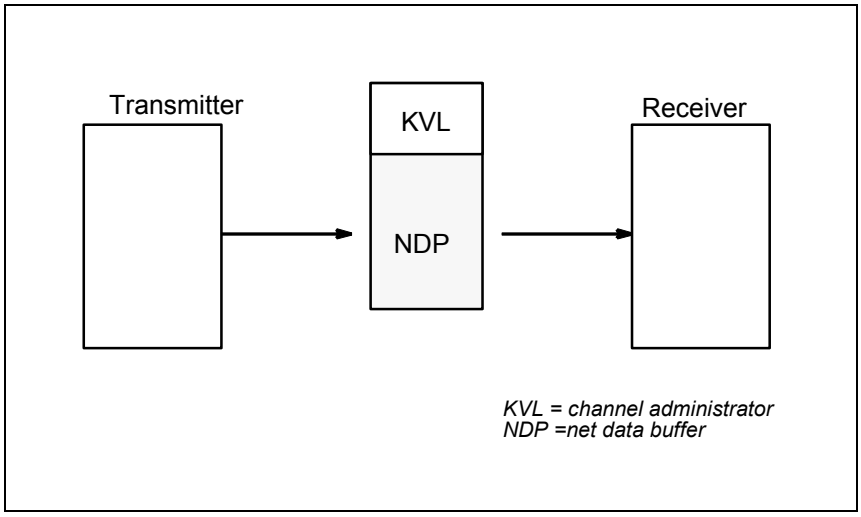


Fig. 3-1 Data transfer in the "Handshake" mode

"Refresh" data transfer mode

The "Refresh" data transfer mode is used,

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

"Refresh" overwrites when it transfers data. The transmitter always deposits the latest data set in the channel without the receiver having acknowledged that it has received the last data set. There are two net data buffers for data transfer, which are used as alternating buffer system. The transmitter flags in which buffer the latest data are located.

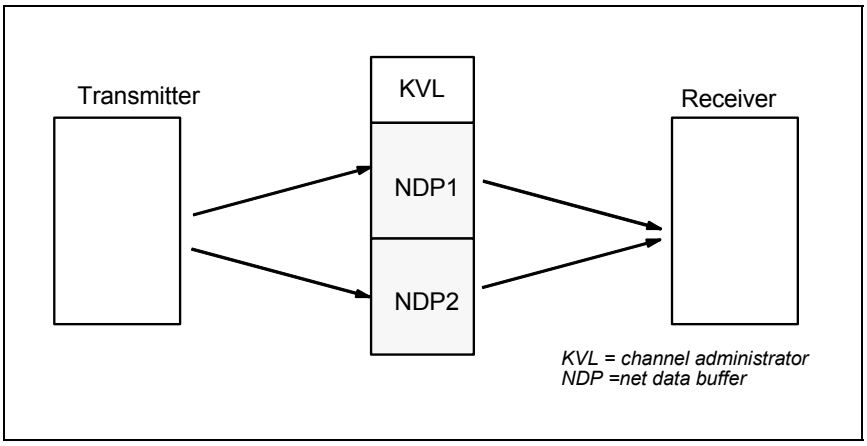


Fig. 3-2 Data transfer in the "Refresh" mode

"Select" data transfer mode

The "Select" data transfer mode is used,

- if information loss may not occur due to data being overwritten, and
- if there can be as many transmitters as required for one receiver

"Select" defines a sequential channel processing. If the receiver acknowledges that it has received the last data set, then the transmitter deposits a new data set in the channel. A net data buffer is provided for data transfer. A channel administrator controls the data transfer.

All of the configured transmitters use the same net data buffer. There is no defined sending sequence for the transmitter. The first one sends first. In order to achieve controlled data transfer, a "1" may only be specified at one transmitter at the EN connection.

The transmitter must be configured in a shorter sampling time than the receiver.

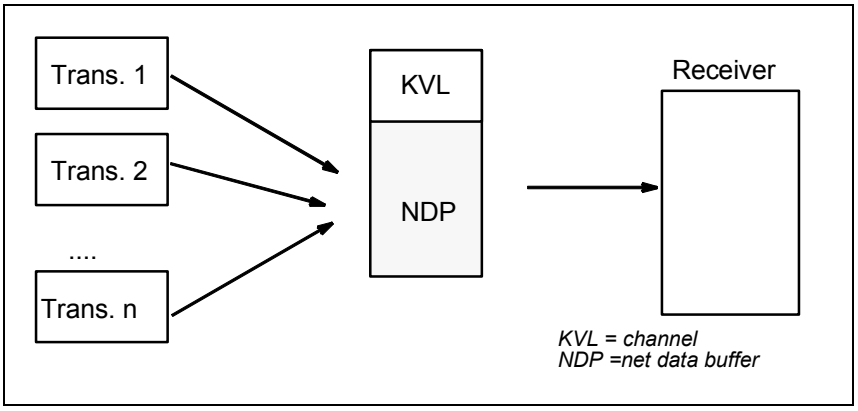


Fig. 3-3 Data transfer in the "Select" mode

"Multiple" data transfer mode

The "Multiple" data transfer mode is used,

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

"Multiple" overwrites data when transferring data. The transmitter always deposits the latest data set in the channel without the receiver first acknowledging that it has received 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 is repeated in the next operating cycle.

There are two net data buffers for data transfer, which are used as alternating buffer system. The transmitter flags in which buffer the latest data are located.

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

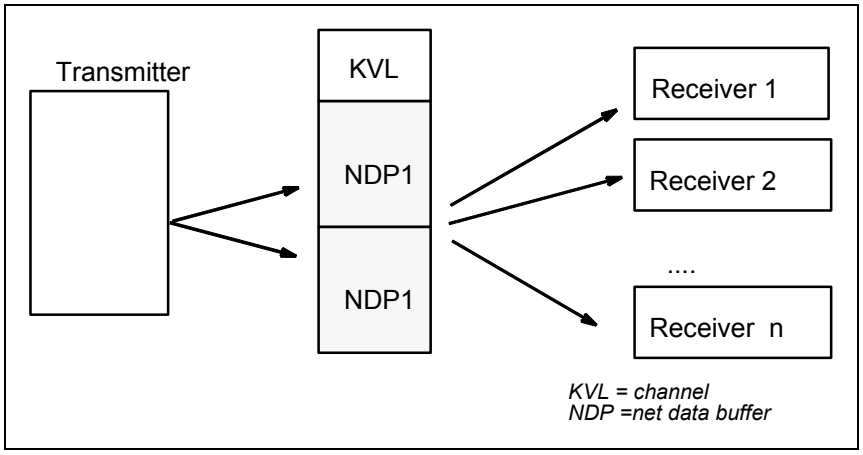


Fig. 3-4 Data transfer in the "Multiple" mode

"Image" data transfer mode

The "Image" data transfer mode" is used for the FM 458-1 DP to communicate via the PROFIBUS DP interface,

- if all of the receivers, which are configured in a task, should be provided with data that come from the same DP cycle,
- if all transmitters, which are configured in the same task, wish to send their data to the DP slaves in the same DP cycle.

To do this, transmitter and receiver FBs synchronize themselves within a task in order to supply consistent data. They form a so-called "**consistency group**". All receiver FBs, associated with such a consistency group, fetch their net (useful) data from a common alternating buffer and all of the transmitter FBs deposit their data in such a buffer.

"Image" is an overwriting data exchange (refer to refresh). There are two net (useful) data buffers which are used for data exchange. They are used as alternating buffer system.

This data transfer mode is only permitted for the PROFIBUS DP interface of the FM458-1 DP application module.

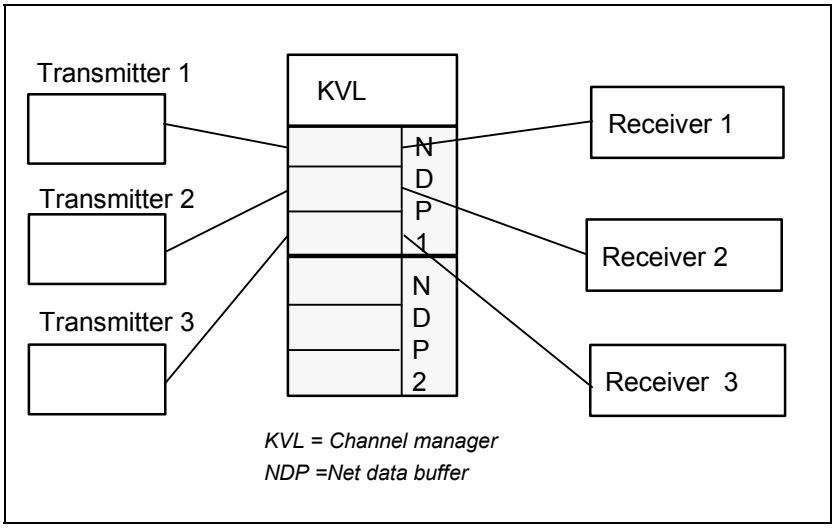


Fig. 3-5 Data transfer in the "Image" mode

3.1.3.4 Firmware status, ECL, ECO connection

General

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

Function

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

- ECL=0 and ECO=0:
  The firmware is in an error-free condition.

- ECL=0 and ECO>0:
  The firmware has an error condition, which can be rectified by the configuring engineer or user. The error cause is described in the Chapters associated with the individual couplings.

- ECL>0 and ECO>0:
  An irreparable firmware error is present.

3.1.3.5 Status display, output YTS

General

The block outputs an error code or the instantaneous data transfer status at its output YTS.

Displayed error

- Real (severe) run-time errors
- Configuring errors, which are identified when the system is initialized, and which are displayed at the 7-segment display of the CPU using a flashing "C".
- Temporary status displays and alarms

Fault diagnostics

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

Additional information
regarding the significance of the decimal number, refer to the online help "Help on events".

3.1.4 Mode of operation of the couplings

General

A coupling functions as follows:

- CPUs transfer data with a coupling module via the backplane bus.
- For serial couplings (e. g. for TCP/IP) the firmware on the coupling module "re-structures" and "packages" the data, so that they correspond to the required telegram structure and protocol.
- If the communications partner is also a SIMATIC TDC (subrack coupling, buffer memory coupling), then the data are not conditioned.

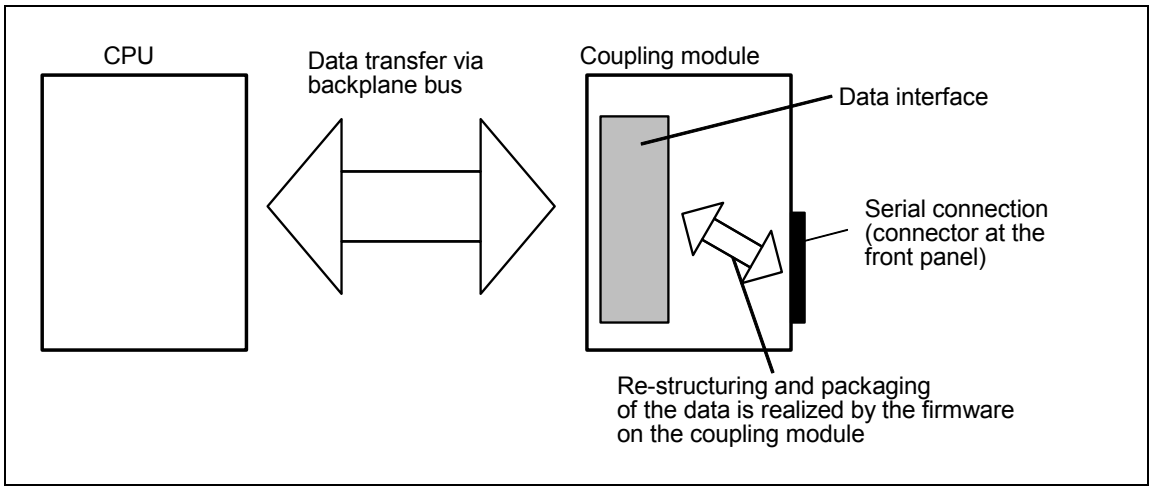


Fig. 3-6 Data transfer between the CPU and coupling module

Access to the data interface

As the data interfaces are located on external coupling modules and not locally on a CPU, they can be used by all CPUs in a subrack. However to use a data interface, the CPU and the coupling module must have the same bus connection.

NOTE

For the local CPU coupling, the data interface is located on the CPU RAM. This data interface cannot be accessed by any of the other subrack CPUs. It can only be used by that CPU on which it was configured.

Basic initialization

A coupling module is always initialized (basic initialization) at system run-up.

The first CPU into the subrack executes the following tasks:

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

Configuring the coupling module

The required coupling module is configured in HWConfig. When initializing a coupling (basic initialization), no explicit configuring steps must be executed.

3.1.4.1 Central coupling blocks

Function of the central coupling blocks

The central coupling blocks have the following functions for a coupling:

- Initialization:
  - copying the configured initialization information (this is configured at the initialization inputs) at the data interface
  - defines as to whether the data interface is in an error-free condition.
- Enabling:

- after initializing 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 subrack. Data transfer can now start.
- for timing reasons, a coupling is always enabled in the RUN condition after several sampling times.

- Monitoring:
  - the central coupling blocks provide information at their outputs about the status of the coupling and, if relevant, the status of the firmware.

Configuring the central coupling blocks

When configuring, the following points must be observed:

- Exactly one central coupling block must be configured for each coupling.
- The central coupling blocks can all be configured on a CPU of a subrack or they can be distributed over various CPUs of a subrack.
  - configuring all central coupling blocks on a CPU simplifies, for example, diagnostics.
- Central coupling blocks have no transmit- or receive functionality.
- All central coupling blocks must be configured in a sampling time 32 ms ≤ TA ≤ 256 ms

Errors

The central coupling block makes an entry into the communications error field and no longer processes the coupling module, if

- a central coupling block identifies an error when being initialized
- a firmware does not respond or manifests erroneous behavior,
- the central coupling block is running on the incorrect communications module.

3.1.4.2 Transmitters and receivers

General

Transmitters and receivers are:

- function blocks, which access the data interface of a coupling, either writing and/or reading.
- part of the communications utility which uses the coupling.

Examples of transmitters:

- message output, function block MSI:
  Copies messages from the message buffer into a data interface
- process data transmit block CTV

Examples of the receivers:

- process data receive block CRV

Data entries at the connections

As transmitters and receivers don't differentiate between the individual couplings, at the block inputs of the transmitter and receiver, a coupling type must not be specified.

I/O, trans./ receivers

- CTS input to specify the coupling module
- address connection AR, AT or US to specify channel names and coupling-specific addresses

Synchronizing transmitters and receivers

Before transmitters and receivers can transfer data, they must first identify- and synchronize with one another:

- Identification is realized via the data configured at the connections CTS and AT, AR or US.
- Synchronization is only possible,
  - if a transmitter identifies its partner as receiver (or vice versa).
  - if the length of the reserved data areas coincide.
  - if the net data structure is compatible.
  - if the data transfer mode is identical (data entry at the MOD input for transmitters/receivers).

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

3.1.4.3 Compatible net data structures

General

The net data structures include information regarding the structure of the net data to be transferred:

- data regarding the position and data types of the associated net data

The net data of the transmitter and receiver must be structured the same to permit data transfer between the transmitter and receiver.

Data types

The following standardized data types are used:

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

Table 3-11 Standardized data types

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

- **Octet string**
  An octet string is an unstructured data type which does not appear at the block I/O (refer to the Chapter Channel marshalling blocks CCC4 and CDC4).

- **Time and date**
  Data type for the time which does not appear at the block I/O (refer to communications utility, message system).

  Value range

  – **1st octet and 2nd octet:**
    Specify the date relative to 1.1.1984.
    Resolution=1 day
    0 days≤d≤65535 days

  – **3rd octet to 6th octet:**
    Specify the time between 00:00 and 24:00.
    Resolution = 1ms
    0 ms≤x≤86400000 ms
    The first 4 bits are not assigned in the sixth octet

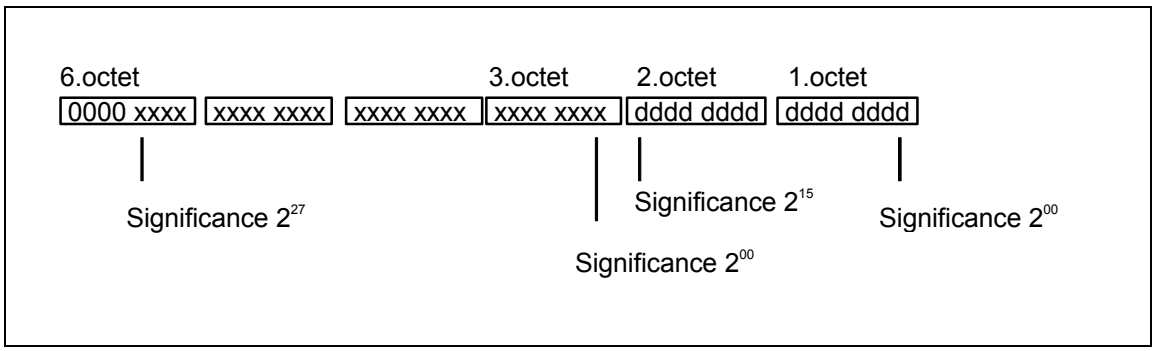


Fig. 3-7 Time and date

3.1.4.4 Number of coupling modules in a subrack

Overview The number of coupling modules (CP50M0, CP51M1, CP5100 and CP52A0) are restricted by two system limits:

- subrack size
  The largest subrack in the SIMATIC TDC system has 21 slots. As a subrack must have at least one CPU, theoretically, 20 slots remain.

- available address space
  In practice, these limit is seldomly reached. For CP modules have 144 Mbyte address space.

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

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

Module type	Occupied address space
CP52A0	2 Mbyte
CP5100	4 Mbyte
CP50M0 CP50M1	2 Mbyte or10 Mbyte buffer memory
CP51M1	1 Mbyte or 9 MByte buffer memory

Table 3-12 Occupied address space

3.1.4.5 Reorganizing a data interface

General

A data interface can be re-formatted without having to interrupt the RUN condition or diminish performance.

Formatting the data interface

Several central coupling blocks have a CDV connection (Bool. data type). If there is a positive edge at the CDV input, the central coupling block inhibits the coupling, and after approx. 10 seconds, formats the data interface. The data interface is then enabled again.

During this inhibit time and while the data interface is being re-formatted, all transmitters/receivers go into a wait condition. After enabling, channels log-on (register) and synchronize just the same as when the system runs-up.

This data is ignored for local data interfaces on a CPU.

3.2 Couplings on the subrack

Overview

These couplings include:

- local CPU coupling
- direct CPU-CPU coupling
- communications buffer coupling

3.2.1 Local CPU coupling

General

The local CPU coupling does not require a coupling module. This coupling type can only be used by function blocks, which are located on the same CPU as the data interface. The data interface always lies on that CPU and is 1 Mbyte.

Application

The coupling is mainly used for autonomous tasks (e. g. a closed-loop control) to provide defined interfaces. Thus, when configuring a project, it is simple if a CPU is "overloaded" to shift the complete task to another CPU without involving extensive configuring work. Communications can, for example, then be realized via the data interface in the buffer. Only the data at the CTS connection has to be changed at all communication function blocks.

Initialization and monitoring

The @LOCAL central blocks cyclically initialize and monitor the coupling. Thus, at the start of cyclic operation, the coupling is not enabled for all senders/receivers, but only after a delay of several operating cycles. The @LOCAL central block monitors the coupling after the coupling has been enabled.

Configuring

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

For the local CPU coupling, only the channel name has to be specified at the AT, AR or US connections of the send/receive blocks. Data for address stages 1 and 2 should not be configured. Transmitters and receivers with the same channel names communicate with one another.

3.2.2 Direct CPU-CPU coupling

General information

The data interface fort he direct CPU-CPU coupling is locally provided on a CPU and has a size of 1 Mbyte.

The data interface must be set-up on the CPU that contains the receive blocks. The reason for this is that the receive blocks cannot access the memories of the other CPUs via the backplane bus. This is only possible from the send blocks.

If the data interface is configured on the send side, this results in a communications error (“C”) at the receive block.

Bild 3-8 Direct CPU-CPU coupling

Use

The direct CPU-CPU-coupling is used to exchange process data between various CPUs of a subrack. Contrary to a coupling memory coupling, higher data quantities are effectively transferred. With this type of coupling, one side (sender) accesses the data interface via the backplane bus – while the others (receivers) can access locally, saving time (fast access).

Initialization and monitoring

A central block @LOCAL should be configured on the CPU on which a data interface should be set-up. This is to initialize and monitor the coupling.

Configuring

The @LOCAL central block initializes and monitors the coupling in cyclic operation.
This means that the coupling is not released for all send/receive blocks when cyclic operation starts – but instead with a delay of several operating cycles. After the coupling has been enabled, the @LOCAL central block monitors the coupling.

The direct CPU-CPU coupling can only be used by send/receive blocks that are configured in the same subrack.

For the direct CPU-CPU coupling, the channel name and in addition,an “E” as address stage 1 should be specified at connections AT or AR of the send/receive blocks. This data is optional for the CPU that must access via the backplane bus (sender).
Data for address stage 2 do not have to be configured. Senders and receivers with the same channel names communicate with one another.

Only “Handshake“ or “Refresh“ are possible as data transfer mode.

3.2.3 Communications buffer coupling

General

The 1 Mbyte data interface for the communications buffer coupling is located on a communications buffer.

The hardware consists of modules CP50M0, CP50M1, CP51M1 or CP53M0.

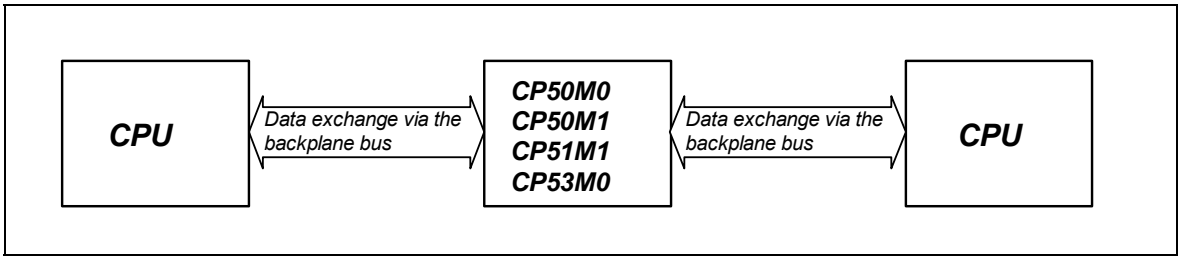


Fig. 3-9 Communications buffer coupling

Application

The communications buffer coupling is used to transfer data between various CPUs of a subrack. Contrary to $ signals, higher quantities of data are transferred more effectively.

Initialization and monitoring

A @GLOB central block must be configured on any CPU of the subrack to initialize and monitor the coupling.

The @GLOB central block cyclically initializes and monitors the coupling. Therefore, the coupling is not enabled for all transmit/receive blocks at the start of cyclic operation but only after a delay of several operating cycles. The @GLOB central block monitors the coupling after the coupling has been enabled.

Configuring

The communications buffer coupling can only be used by send/receive blocks which are configured on the same subrack.

For the communications buffer coupling, only the channel name has to be specified at the AT-, AR- or US connections of the transmit/receive blocks. Address stages 1 and 2 do not have to be configured. Transmitters and receivers with the same channel names communicate with one another.

3.3 Subrack coupling CP52M0

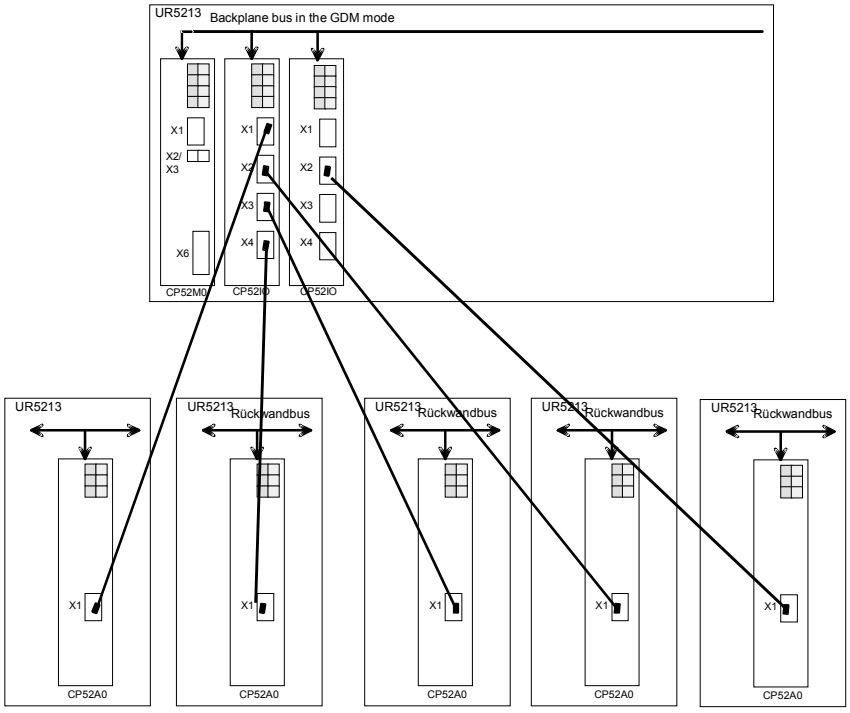
3.3.1 Applications

Data can be exchanged, across all of the subracks, between all CPU modules in the system via the memory in the **G**lobal**D**ata**M**emory (**GDM**).

Up to 44 subracks can be coupled synchronously via the central memory. This means that 836 CPU modules can be used when the system is expanded up to its maximum.

The GDM has its own dedicated subrack (known as the **GDM subrack** in the following). The **CP52M0 memory module (slot 1)** and an appropriate number of **CP52IO interface modules (slots 2-12)** are accommodated in the UR5213 subrack with 21 slots.

Each subrack, which is coupled with GDM (these are called coupled subracks in the following text), must have a **CP52A0 access module**. These subracks are connected in a star configuration to GDM via glass-fiber optical cables.



((in diagram: Rückwandbus = Backplane bus))

Fig. 3-10 Example of a GDM system with 5 subracks coupled through fiber-optic cables

CP52M0 function

The 2Mbyte central memory of the GlobalDataMemory system is located on the CP52M0 GDM memory module. This central memory handles all of the data transfer between the processes in the coupled subracks.

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

Literature on this subject:

- GDM hardware documentation

3.3.2 Behavior when powering-up and powering-down

- The subracks can be powered-up in any sequence.
- If the GDM subrack is powered-down in operation, then all of the telegram channels are disabled in the coupled subracks. These telegram channels are re-initialized after the GDM subrack has been powered-up again.

NOTE If the GDM subrack is powered-up while the coupled subracks are running, then an increased computational time can be expected when establishing a link for the CPUs communicating via the subrack coupling. For CPUs, which are already heavily utilized, this can result in an 'E' being displayed (error in the task manager).

- All of the coupled subracks can be powered-down and up again while operational. The coupled partners can then no longer receive data from this subrack nor send data to it.
- If a coupled subrack is powered-down and up again, then the communications between the remaining nodes (GDM subrack and a maximum of 43 coupled subracks, each with a CP52A0) are not affected.
- Coupled subracks which are powered-down can be re-configured and then powered-up again. Even if the number of senders and receivers has been changed (e.g. if one sender too little was configured).

NOTE If the data length is changed and a coupling exists, then this results in communication errors in the subrack where the configuring was changed. In this particular case, the GDM subrack must also be reset.

- As soon as the subrack which was powered-down is powered-up again, a new coupling is established between the coupled partner which was powered-up and the GDM subrack. The telegram channels are re-initialized and communications are re-established without having to reset the coupled partner.

3.3.3 Synchronizing and triggering methods

The following synchronizing and trigger signals are available on the CP52A0 access module:

1. Basic clock cycle
2. Bus interrupt
3. Clock time

The clock time and basic clock cycle synchronizing signals can be sent, independently of one another only from one CP52A0. They can either be sent or received on a CP52A0. The decision to send or receive (basic clock cycle) or only to send (clock time) is made in HWConfig.

If the clock time and the basic clock cycle are configured for sending on several access modules, the signal is always sent from the CP52A0, which is connected with a CP52IO, whose fiber-optic cable interface is located to the farthest left in the subrack.

The SYSFAIL* and bus alarm signals can be simultaneously sent and received from every CP52A0.

3.3.4 Configuring

Rules

1. The complete user application software is made in the coupled subracks.
2. The names of the coupled subracks (UR5213) must be assigned **unique** names (under the "General" tab of the object properties dialog box).
3. A CP52A0 GDM access module should be configured as communications module for each coupled subrack at any slot in HWConfig.
4. Only **one** CP52A0 GDM access module may be configured per coupled subrack.
5. A @SRACK central block must be configured for each CP52A0.

3.3.4.1 Configuring in HWConfig

The hardware configuration of a subrack is defined in HWConfig. The CP52A0 module is in the SIMATIC TDC module catalog under communication modules.

Under the “Synchronizing” register tab, you can specify whether the basic clock cycle, bus alarm and clock time signals are sent, received or should be de-activated:

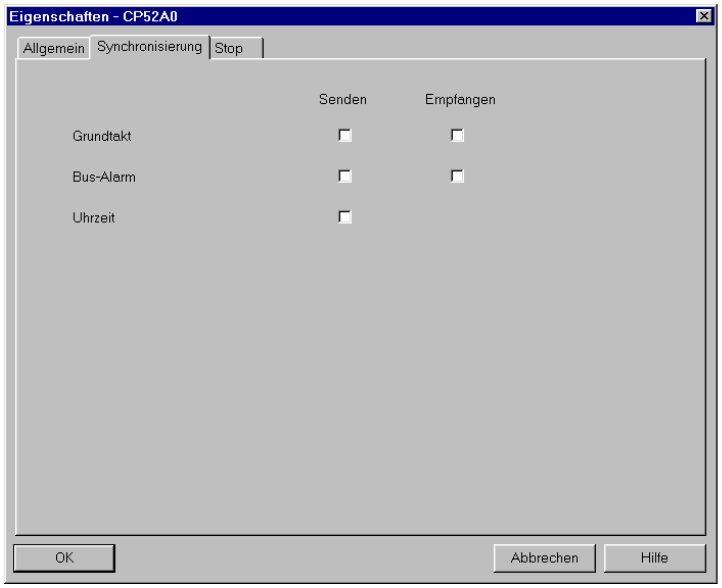


Fig. 3-11 "Synchronizing" tab

The **basic clock cycle** signal can have the following statuses:

1. Receive signal, **or**
2. Send signal **or**
3. Inactive

The following statuses are available for the **bus alarm** signal:

1. Receive signal, **and / or**
2. Send signal **or**
3. Inactive

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

1. Send signal, **or**
2. Inactive

NOTE The reception of the clock signal is set at block RTCM.

The signals are de-activated, if none of the buttons are selected.

Default setting The signals are switched **inactive** (not selected) as **default setting**. The user can specify, for each signal, whether he wishes to send it, receive it or wishes to keep the basic setting, by simply clicking on the button with the mouse.

Under the "Stop" tab, it is possible to set whether the module should stop the complete rack if the system error signal (SYSFAIL*) is received from other subracks. It can be simultaneously determined whether this signal is to be sent.

NOTE As long as a coupled subrack flashes with "H" (SYSFAIL active), and this rack sends as a result of the SYSFAIL signal configuring, then the other coupled subracks no longer run-up. In this case, the master subrack must be powered-down and the coupled subrack, which caused the SYSFAIL signal, must be reset.

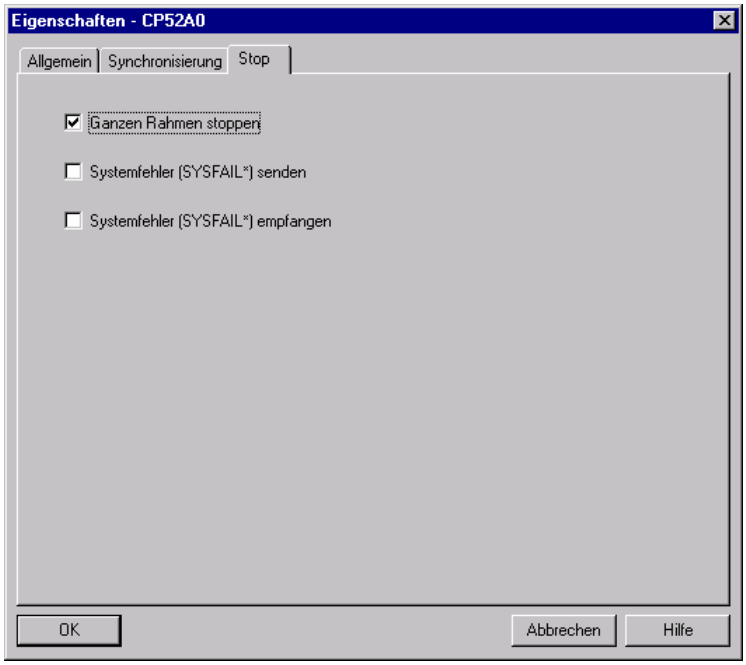


Fig. 3-12 "Stop" tab

3.3.4.2 Configuring in CFC

The @SRACK central block must be configured for the CP52A0 module of the coupled subrack.

In this case, the "**CTS**" connection (CP52A0 module name) must be configured.
Starting from N01 (and then increasing consecutively, without any gaps, with N02 etc.), it is possible to specify at the "Nxx" connections (name, subrack xx), which subrack should be monitored. The block evaluates the connection assignment until the first connection, with an empty string.

The data transfer is configured using send and receive blocks. For the subrack coupling, only the channel name is specified at the AT and AR inputs. Data for address stages 1 and 2 don't have to be configured. Senders and receivers with the same channel names communicate with one another.

3.3.5 Performance data

3.3.5.1 Data transfer rates

The data transfer rate between the individual subracks is very dependent on the configuration.

For a configuration of approx. 20 subracks, each with 4 CPUs, a **60 bytes/ms data transfer rate** can be assumed per CPU when sending and receiving.

3.3.5.2 Cable lengths

The performance of the subrack coupling depends on the length of the fiber-optic cables used. Thus, we recommend that the shortest possible cables are used. Longer cables slow down the memory access and, on one hand increase the computation loading of the CPU, and on the other hand, they reduce the maximum possible data transfer rate.

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

3.3.5.3 Interface assignment

The last assigned interface for each CP52IO provides a somewhat wider time window for the GDM access. This results in, for the same data rate, a lower computational time loading of the CPU550.

Example

Seven subracks are connected to the GDM subrack via two CP52IO.

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

This means that interfaces **X4** (first CP52IO) and **X3** (second CP52IO) have a lower CPU computational load on the connected subracks.

3.4 Subrack coupling CP53M0

General

The following couplings can be realized using the CP53M0 module:

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

- SIMADYN D as master:

At any location, instead of a CS22, a CP53M0 can be connected to the CS12/CS13/CS14. The CP53M0 should be parameterized in the slave mode.

- SIMATIC TDC as master:

A CP53M0 is parameterized in the master mode in a SIMATIC TDC subrack. A CS22 or a CP53M0 in the slave mode can be inserted at the two fiber-optic cable interfaces.

In addition to the coupling to SIMADYN D, up to three SIMATIC TDC subracks can be coupled with one another. The CP53M0 is parameterized in the master mode in one of the subracks; the CP53M0 is parameterized in the slave mode in the other two subracks.

In the following text, the subrack in which the CP53M0 module is in the master mode, is designated as the master subrack.

The subrack in which the CP53M0 module is inserted in the slave mode, will be designated as the slave subrack.

The coupling via a CP53M0 in the master mode or slave mode is configured just like a coupling on the CS12 or CS22 on the SIMADYN D side.

Initialization and monitoring

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

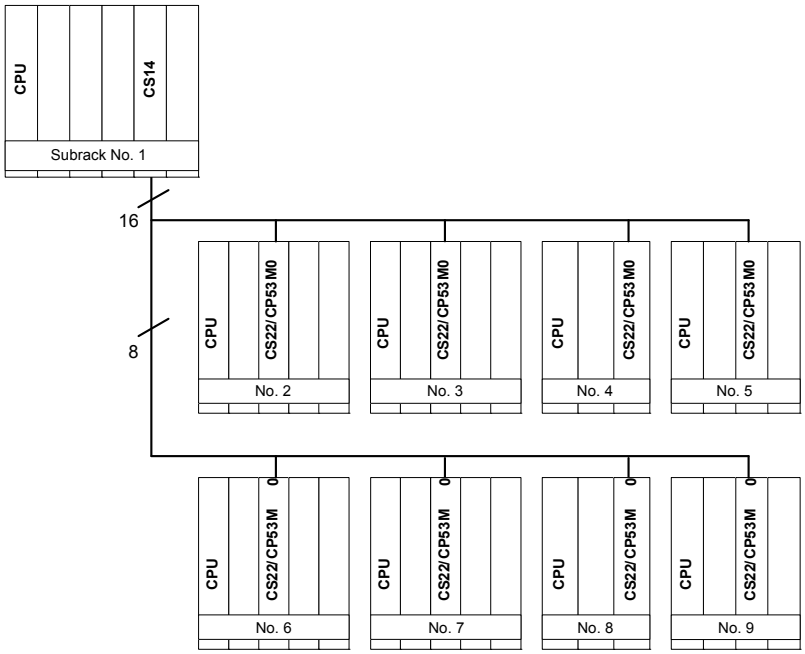


Fig. 3-13 Maximum configuration for 8 Slaves with CS14

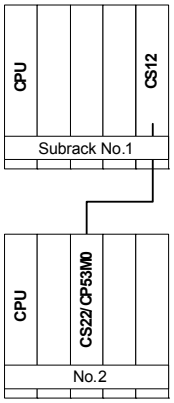


Fig. 3-14 Point-to-point coupling with CS12

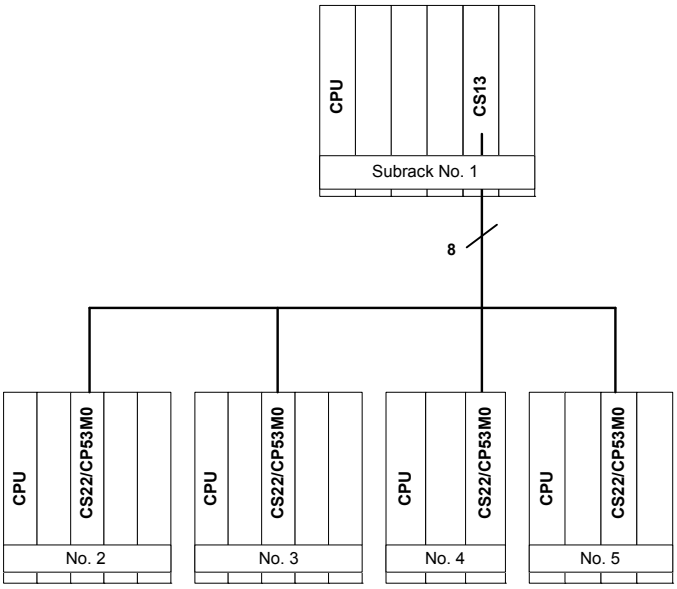


Fig. 3-15 Configuration for four slaves with CS13

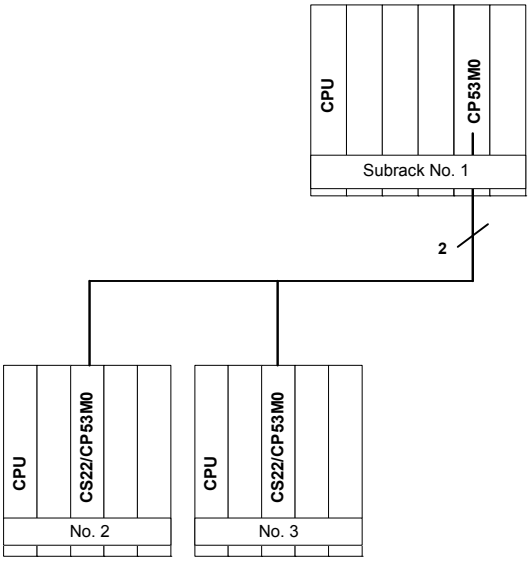


Fig. 3-16 Maximum configuration with CP53M0 as master

3.4.1 Hardware structure

Overview

- Only SIMADYN D subracks with C-bus connection can be coupled on the CP53M0 (e. g. SR24).
- The master subrack, has, depending on the number of slaves to be connected, a CS12, CS13, CS14 or a CP53M0 module. The slave subracks have a CS22 or a CP53M0 module.
- A maximum of 2 slave modules (CS22 or CP53M0 in the slave mode) can be coupled to a CP53M0 in the master mode.
- Several CP53M0 in either the slave mode or master mode can be configured in a SIMATIC TDC subrack. Thus, several different subrack couplings can be configured in a subrack.
- The CP53M0 module of a subrack coupling must all be configured in different subracks.

3.4.2 Scope of supply

Overview

All of the slave subracks are permanently coupled to the master subrack, as a slave subrack must continuously access the memory in the master subrack.

- The master/slave subracks can be powered-up in any sequence.
- All subracks can be powered-down and up again in continuous operation.
- If a slave subrack is powered-down and up again, then communications between the other nodes is not influenced.
- Slave subracks which are powered-down can be re-configured and powered-up again. The number of transmitters and receivers can also be changed (e. g. if one transmitter too little was configured).
- As soon as the slave partner, which was powered-down, is powered-up again, a new connection is established between the new partner which has been powered-up again 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 transfer area and not as communications partner. Slave-slave communications are interrupted when the master subrack is powered-down.

NOTE It is not permissible to remove the fiber-optic cable during operation as this can result in a CPU crash.

3.4.3 Response when "shutting down" a coupling partner

Response of the master subrack

The master subrack is shutdown:

The @CS2 central block can no longer access the master subrack and prepares a restart (in addition, the CDM block output is set to "faulted", refer to @CS2 mask). The system then waits until the master subrack is powered-up.
All slave transmit/receive blocks can no longer access the master subrack and start a new channel log-on.

Response of the slave subrack

The slave subrack is shutdown:

The @CS1 central block and the maximum seven additional @CS2 central blocks decrement their particular NCP connection (i. e. the number of active slave subracks is reduced by one). Otherwise, there is no response, and the NCP connections are incremented again after the appropriate slave subrack runs-up again.
All of the configured transmit/receive blocks, whose coupling partner is located on the subrack which is shutdown, wait until the subrack has run-up again.

3.4.4 Response when "powering-up" the master subrack

Response

If the master subrack is powered-up again while the slave subracks are operational, it can be assumed, that for a short period of time, increased computation time will be required to establish the connection for CPUs to communicate via the subrack coupling. For already highly utilized CPUs, this can result in an 'E' being displayed at the 7-segment display (error in the task administrator).

3.4.4.1 Acknowledging

The 'E' can be acknowledged in two ways:

Manual acknowledge

When manually acknowledging, after the connection has been established, the 'E' can be acknowledged by depressing the red acknowledge button on the CPU.

Automatic acknowledgement

For automatic acknowledgement, the following must be configured on all CPUs in the slave subrack which communicate via the subrack coupling. Automatic acknowledgement can be implemented in two different ways using this particular configuration:

1. All YEV outputs, of the function blocks communicating via the subrack coupling are monitored using a software which has to be configured. If the value of all YEV outputs is less than 9 (i. e. initialization has been completed), then the input NOT.I is set to '1'. Using the CDM output of the @CS2 central block, it is ensured that the system is only automatically acknowledged if the master subrack has actually been powered-up. Using the time limit (input T at PCL), automatic acknowledgement has to be realized within a certain time. The 'E' on the 7-segment display is now automatically acknowledged using the SYF4 function block.

2. If not all of the YEV outputs can be monitored or should be monitored, input OR.I2 should be set to "1" and input NOT.I should not be connected at all. In this case, the CPU is only acknowledged within the time, set at connection T of the PCL after the master subrack has been powered-up (output of @CS2.CDM).

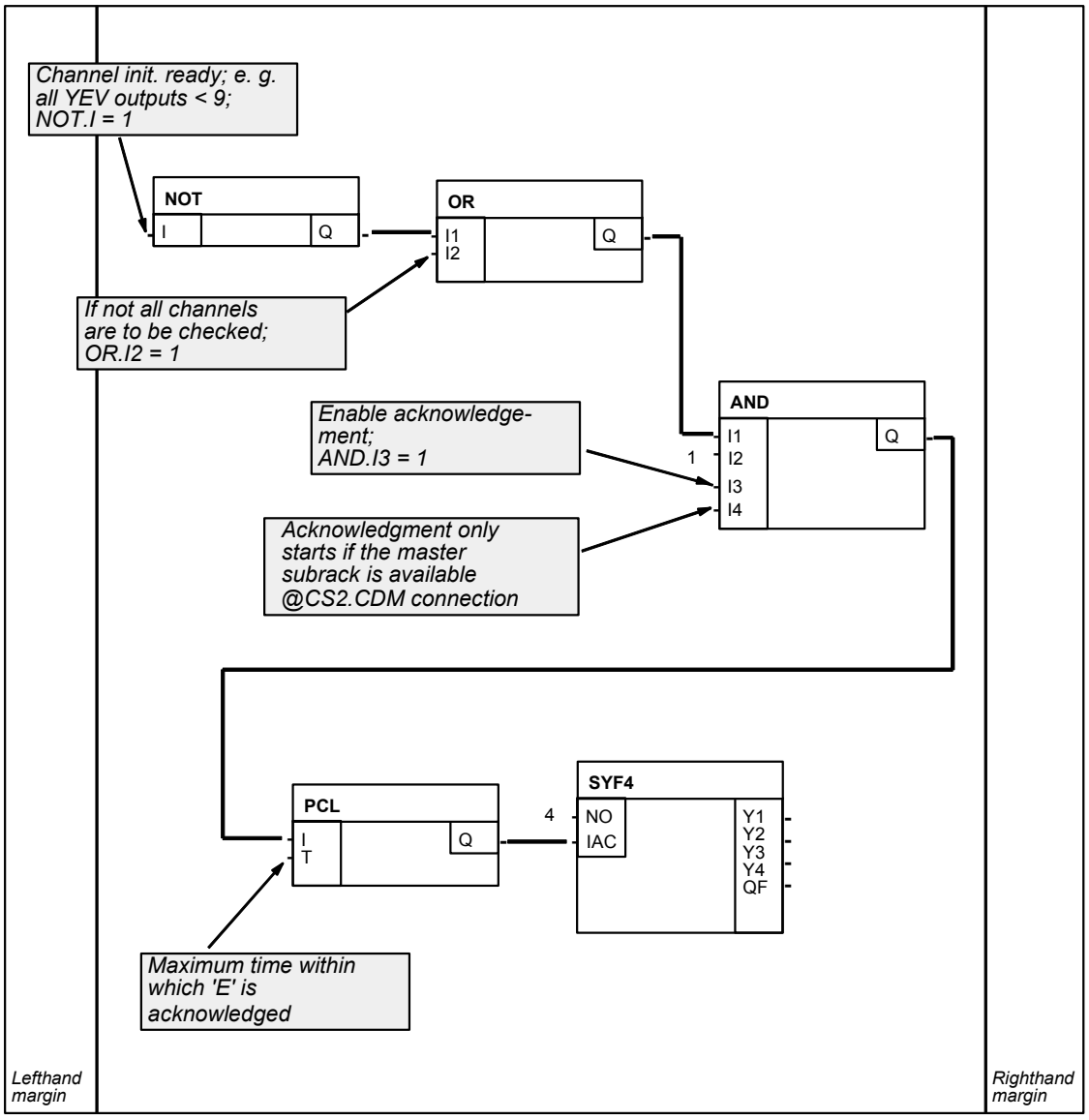


Fig. 3-17 Automatic acknowledgement of 'E'

3.4.5 Restart capability

Synchronizing transmitters and receivers

An additional important communications feature for external communication interfaces is the restart frequency of transmitters/receivers. Transmitters/receivers always re-locate their old channel and re-synchronize with them.

Subracks can be powered-up and down again in any sequence. The transmitters/receivers of the subracks, in which the CP53M0 (slave mode) is inserted, synchronize themselves to the previous channelsat each restart (new run-up).

If a transmitter/receiver identifies the "right" channel at log-on, then it cannot identify

- if it had previously used this channel before.
- whether this channel is presently being used by another transmitter/receiver (or transmitter or receiver).

3.4.6 Configuring

Rules

- For a fiber-optic cable subrack coupling, all of the CP53M0 modules (slave mode) must have different names. If names have been assigned twice, then the appropriate central blocks log-off with multiple configuring (FB disable).
- All CP53M0 modules of a subrack must be inserted in different subracks.
- The sampling time range, 32 ms≤TA≤256 ms, valid for central coupling blocks, is also valid for the subrack coupling central blocks @CS1 and @CS2. It should also be observed that the @CS2 central blocks may only be configured, as maximum, in twice the sampling time as the @CS1 central block (in the case of basic clock synchronization also same sampling times are permissible).
  The actual sampling time (in milliseconds) is decisive and not the cyclic task (T1, T2 etc.)
  - Example: If the @CS1 central block was configured in 100ms, the @CS2 central blocks can be configured in a sampling time up to 200ms (180ms, 150ms, 130ms, 50ms etc. are thus permitted).

Data interface

The data interface is located on the dual port RAM of the CP53M0 module (maste rmode). The available data transfer area is 128 kbytes.

Initialization and monitoring

The coupling initialization and monitoring is handled by the @CS1 and @CS2 central blocks in the RUN status. Thus, the coupling is not enabled at the start of cyclic operation for all transmit/receive blocks, but is delayed by several sampling cycles. The coupling is always first enabled in the master subrack and then in the slave subracks.

After the coupling has been enabled, central blocks @CS1 and @CS2 monitor the coupling. In this case, the number of active coupling partners is output at the central block outputs.

Names at the AT- and AR inputs

For the subrack coupling, only the channel name has to be specified at the AT- and AR inputs of the transmit/receive blocks. Names should not be configured for address stages 1 and 2. Transmitters and receivers with the same channel names communicate with one another.

3.4.7 Restrictions

Attention should be paid to the following restrictions, if data are exchanged with a SIMADYN D rack configured with STRUC:

- Only the blocks CTV / CRV may be used in configuring SIMATIC TDC. The use of blocks CTV_P / CRV_P (pointer blocks) causes a communications error (‘C’).
- A data exchange can only take place with data types I2, I4 and Real (NF) in STRUC. All the other data types cause a communications error (‘C’). You can use type switching in STRUC in this case, though.

You should also pay attention to the different access times to the module CP53M0 over the backplane for master and slave:

- Master: approx. 1 µs (4 Byte)
- Slave: approx. 8 µs (4 Byte)

The load of the CPUs in the slave rack is proportionately increasing using the same number of data.

3.5 TCP/IP coupling (CP51M1)

Introduction

This application software example is intended to show how a design engineer should proceed when implementing a SIMATIC TDC subrack coupling via TCP/IP or UDP.

The configuration described here includes the basic hardware equipment as well as function blocks and shows how it is used. A conscious decision was made to keep the functional scope of this application software extremely low, so that those reading it can get up to speed quickly on the subject. The functionality and/or the hardware components can be expanded at any time. However, the information in the applicable function block documentation must be observed.

All of the names used in this application software have been randomly selected and are only binding for this configuration example.

The structure of these configuring instructions essentially determines the sequence that the various steps are made with which the complete application software can be generated. However, this should only be considered as a recommendation and does not have to be rigidly maintained.

Application cases

CP51M1 can be used for the following applications:

- Exchanging process data with other CP51M1 / CP5100 and SIMATIC Industrial Ethernet modules (e.g. CP443-1)
- Visualizing process data using WinCC
- Visualizing messages using WinCC
- Exchanging process data with third-party systems (e.g. process computer)
- Central commissioning (start-up) and diagnostics of all CPU modules in the subrack
- Time synchronization to use a unified time within a particular plant or system
- CP50M1 from FW version V1.1 onwards and with D7-SYS from V7.0 onwards supports the routing function.

NOTE

When integrating new modules into an existing TCP/IP network, you should always ask your network administrator about the IP, sub-network and router addresses as well as the port numbers for applications.

Literature on this subject:

- TCP/IP basics
  (e.g. W. Richard Stevens, TCP/IP Illustrated, Volume 1, Addison-Wesley Verlag)
- CP51M1 hardware documentation

3.5.1 Comparison between TCP/IP and UDP

TCP/IP

The TCP/IP protocol type is connection-oriented. This means that data can only be sent if the coupling partner can also be addressed. As long as there is a connection to the coupling partner, the TCP/IP protocol ensures that the data which is sent also arrives at the coupling partner. If a fault condition develops, the data are, if necessary, transferred several times.

If a connection is only established at one end, **data could be lost** in the TCP/IP stack of the CP51M1 or the coupling partner (e.g. PC) as a result of the protocol.

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

UDP

The UDP protocol type is not connection-oriented. This means that data is also sent even if the coupling partner is temporarily not addressable. **Data is then lost**. However, received data is correct as a result of the data security mechanisms.

When compared to TCP/IP, UDP can achieve a higher communications speed between the coupling partners. However, this is only possible with a somewhat restricted data transfer security.

Channel modes

The following channel modes are possible using the TCP/IP and UDP protocols:

- 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). Thus, for these two channel modes, **UDP** is especially suitable as data transfer protocol.

The **Handshake** and **Select** channel modes are not overwriting; in this case **TCP/IP** should be recommended as data transfer protocol.

Client or server

An overview as to when SIMATIC TDC operates as TCP/IP client or server is provided in the following table:

SIMATIC TDC (TCP/IP) process data (CRV/CTV)		Communications partner
Information of IP address and port in address stage 2	Connect → ← accept Send data →	must be server
no information in address stage 2	← Connect accept → ← Send data	must be client

3.5.2 Typical configuration

The following hardware components are required per station as a minimum for a coupling from SIMATIC TDC via TCP/IP:

- **UR5213** subrack
- **CPU551** CPU module with **MC5xx** memory module
- CP51M1

NOTE A maximum of four CP51M1 can be operated in parallel in one subrack.

3.5.3 Configuring steps

Only the configuring steps for one station are shown in the following. All of the other stations can be essentially handled in the same way.

3.5.3.1 Configuring in HWConfig

Parameterizing the CP51M1

The CP51M1 is parameterized via the appropriate tab in the object property dialog box.

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

- **IP address ("Parameter" tab)**
  IP address of the module in the "dot" notation, in this case: **141.20.135.197**
- **Sub-network mask ("Parameter" tab)**
  Sub-network mask to designate the network segment, in this case: **255.255.0.0** for a Class B network
- **IP address of the default router ("Parameter" tab)**, in this case: **"Use no router"**

NOTE It must be ensured that the IP address and sub-network mask are harmonized. For example, for a Class C IP address (193.x.y.z), the Class B sub-network mask 255.255.0.0 is not permissible. Detailed information on how to select the IP address and its value ranges is provided in the online help for CP51M1.

The following diagram shows the settings, specified above under the "Parameter" tab:

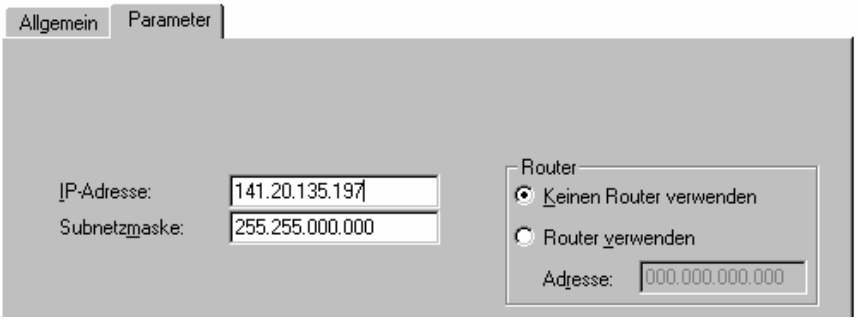


Fig. 3-18: CP51M1 network settings

3.5.3.2 Configuring with CFC

Only the most important I/O are handled in the following. Deviations, which are obtained when configuring a UDP coupling, are described at the appropriate locations.

FBs required

The following function blocks are required for this example:

- **@TCPIP** central block to initialize and monitor the CP51M1 module **(this is always required)**
- **CRV** receive block (in this case, optional, for the process data coupling)
- **CTV** send block (in this case, optional, for the process data coupling)

3.5.3.2.1 Central block @TCPIP

Configuring

Connection	Connection assignment (significance)
CTS	**D1800C.X01** (module name and connector of the configured CP51M1)

3.5.3.2.2 Receive block CRV

"AR" connection

Initialization connection to enter an address. **Address stage 1 and address stage 2 (only when client)** must be specified in addition to the channel name.

The two address stages are separated in the notation by a "**.**".

Rules for address stage 1:

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

Only ports 1024 to 65535 should be used as port number. This should be harmonized with the system administrator. Port numbers up to and including 1023 are generally reserved for “well known services” and “unix-specific services”.

Rules for address stage 2:

- The first **12 digits** define the IP address of the remote coupling partner. This is entered in the so-called “dot" notation; however without specifying the separating point. Leading zeros should be specified (in this case: "**141020135198**" for IP address 141.20.135.198).
- The 13th character must be a “**-**”.
- The port number of the remote coupling partner is defined by the next **5 digits**; leading zeros should be specified (in this case: "**01024**" for port number 1024).

Configuring

Connection	Connection assignment (significance)
CTS	**D1800C.X01** (module name and connector of the configured CP51M1
AR	**RXKAN1.T-01024** (address parameter, receive)
MOD	**H** (receive mode)
EN	**1** (enable)

3.5.3.2.3 Send block CTV

"AR" connection

Initialization I/O for address data. **Address stage 1** and **address stage 2 (only when client)** must be specified here, in addition to the channel names.

The two address stages are separated in the notation by a "**.**".

Rules for address stage 1:

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

Only ports from 1024 to 65535 should be used as port number. Port numbers up to and including 1023 are generally reserved for “well known services” and “unix-specific services”.

Rules for address stage 2:

- The first **12 digits** define the IP address of the remote coupling partner. This is entered in the so-called “dot" notation; however without specifying the separating point. Leading zeros should be

specified (in this case: "**141020135198**" for IP address 141.20.135.198).

- The 13th character must be a "**-**".
- The port number of the remote coupling partner is defined by the next **5 digits**; leading zeros should be specified (in this case: "**01024**" for port number 1024).

Configuring

Connection	Connection assignment (significance)
CTS	**D1800C.X01** (module name and connector of the configured CP51M1
AR	**TXKAN1.T-01024.141020135198-01024** (address parameter, send)
MOD	**H** (send mode)
EN	**1** (enable)

3.5.4 Application information

3.5.4.1 Channel number

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

The actual possible number of channels depends on the size and the number of net data and the access mechanism (handshake, refresh). 254 Kbytes of RAM on the CP51M1 are available for the data interface.

Calculation

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

- **per refresh/multiple channel:** 150 bytes + 2 * number of net data bytes (size of the virtual connection)
- **per handshake/select channel:** 150 bytes + 1 * number of net data bytes

3.5.4.2 Telegram length

The telegram length is restricted as follows:

- To 32767 bytes for receiving
- 32767 bytes for sending

3.5.4.3 "Ping" on CP51M1

Communications with the CP51M1 can be checked using a "Ping". The module only responds to a "Ping" if the communications partner is located within the particular network or a default router (within the particular network) can establish the coupling.

3.5.4.4 Performance

The performance of the 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).

Prerequisites

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

Results

The number of send/receive tasks per CP51M1 is approx. **1270**.

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

Data transfer rate [kbytes/s]	Tasks/s	Send/receive cycle [ms] / connections	Telegram length [bytes]
302.260	1270	64 / 40	192
974.556	911	64 / 56	1024
888.160	208	64 / 16	4096

3.5.5 Communications with WinCC

There are two possibilities of communicating with WinCC:

- Standard coupling (without any supplementary software in WinCC, only process data can be accessed)
- Coupling via TDC PMC TCP channel-DLL (additional channel-DLL for WinCC, access to process data and messages)

3.5.5.1 Standard coupling

When configuring the system on the TDC side, you can proceed as described for MPI in Chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**. The difference is that on the WinCC side, instead of an MPI connection, an Industrial Ethernet connection must be set-up.

NOTE Connections in the system may only be accessed by appropriately addressing data blocks. It is not permissible to access using flags; in WinCC this would result in an error message!

3.5.5.2 Coupling via TDC PMC channel-DLL

When configuring the system, you should proceed as described in Chapter 3.17.

D7-SYS is also supplied with a project example for the CP5100 ("D7-TDC-WinCC") with the appropriate documentation ("D7-SYS – SIMATIC TDC WinCC coupling") - that must be adapted for the CP51M1 (according to 3.5.8).

3.5.6 Central service

Chapter 3.13 describes how to proceed in order to set-up central service (e.g. CFC Online) via CP51M1.

3.5.7 Clock time synchronization

The subject of clock time synchronization is described in Chapter 3.14.

3.5.8 Changing-over from CP5100 to CP51M1

The following procedure should be carefully observed when configuring using the CP5100 on the new CP51M1 module:

- Delete CP5100 in HWConfig and configure CP51M1 (in so doing ensure that the same data is specified as for CP5100, e.g. IP address).

- In the CFC charts, the name of the CP5100 must be replaced at all CTS connections of CTV/CRV blocks using the module name of the CP51M1 and the connector designator (e.g. D1800C by D1800C.X01). The user is supported in doing this by opening any chart for each CPU and calling Options --> Convert CTS connection. Now, only the two names have to be specified in the dialog box that is displayed (available at CTS connections and new ones to be entered). If the names are correctly entered and can be replaced then the names are replaced in the complete chart container. If an error occurs, then an appropriate error message is output.

- Manual changes are only necessary if UDP telegrams are configured with a length > 2048 bytes. For UDP, the CP51M1 can only transfer 2048 bytes!

3.6 TCP/IP coupling (CP5100)

Introduction

This application software example is intended to show how a design engineer should proceed when implementing a SIMATIC TDC subrack coupling via TCP/IP or UDP.

The configuration described here includes the basic hardware equipment as well as function blocks and shows how it is used. A conscious decision was made to keep the functional scope of this application software extremely low, so that those reading it can get up to speed quickly on the subject. The functionality and/or the hardware components can be expanded at any time. However, the information in the applicable function block documentation must be observed.

All of the names used in this application software have been randomly selected and are only binding for this configuration example.

The structure of these configuring instructions essentially determines the sequence that the various steps are made with which the complete application software can be generated. However, this should only be considered as a recommendation and does not have to be rigidly maintained.

Application cases

CP5100 can be used for the following applications:

- Exchanging process data with other CP5100 and SIMATIC Industrial Ethernet modules (e.g. CP443-1)
- Visualizing process data using WinCC
- Visualizing messages using WinCC
- Exchanging process data with third-party systems (e.g. process computer)

NOTE When integrating new modules into an existing TCP/IP network, you should always ask your network administrator about the IP, sub-network and router addresses as well as the port numbers for applications.

Literature on this subject:

- TCP/IP basics
  (e.g. W. Richard Stevens, TCP/IP Illustrated, Volume 1, Addison-Wesley Verlag)

CP5100 hardware documentation

3.6.1 Comparison between TCP/IP and UDP

TCP/IP

The TCP/IP protocol type is connection-oriented. This means that data can only be sent if the coupling partner can also be addressed. As long as there is a connection to the coupling partner, the TCP/IP protocol ensures that the data which is sent also arrives at the coupling partner. If a fault condition develops, the data are, if necessary, transferred several times.

If a connection is only established at one end, **data could be lost** in the TCP/IP stack of the CP5100 or the coupling partner (e.g. PC) as a result of the protocol.

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

UDP

The UDP protocol type is not connection-oriented. This means that data is also sent even if the coupling partner is temporarily not addressable. **Data is then lost**. However, received data is correct as a result of the data security mechanisms.

When compared to TCP/IP, UDP can achieve a higher communications speed between the coupling partners. However, this is only possible with a somewhat restricted data transfer security.

Channel modes

The following channel modes are possible using the TCP/IP and UDP protocols:

- 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). Thus, for these two channel modes, **UDP** is especially suitable as data transfer protocol.

The **Handshake** and **Select** channel modes are not overwriting; in this case **TCP/IP** should be recommended as data transfer protocol.

Client or server

An overview as to when SIMATIC TDC operates as TCP/IP client or server is provided in the following table:

SIMATIC TDC (TCP/IP) process data		Communications partner
Function block CTV (client)	Connect → ← accept Send data →	Receiver
Function block CRV (server)	← Connect accept → ← Send data	Sender

3.6.2 Typical configuration

The following hardware components are required per station as a minimum for a coupling from SIMATIC TDC via TCP/IP:

- **UR5213** subrack
- **CPU550** CPU module with **MC5xx** memory module
- **CP5100** communications module (here at slot 18)

NOTE

CP5100 may only be configured at **slots 18 to 21**. This means that a maximum of four CP5100 modules can be operated in parallel in a subrack.

Depending on the selected slot, BCD coding switches S1 to S3 (refer to Fig. 3-19 for their location on the module) must be set on the module according to the table below. The combination “C-8-0” is obtained for the typical configuration.

Slot	S1	S2	S3
18	C	8	0
19	C	8	4
20	C	8	8
21	C	8	C

((Fig. below: Coding switches to adapt to the slot in the subrack))

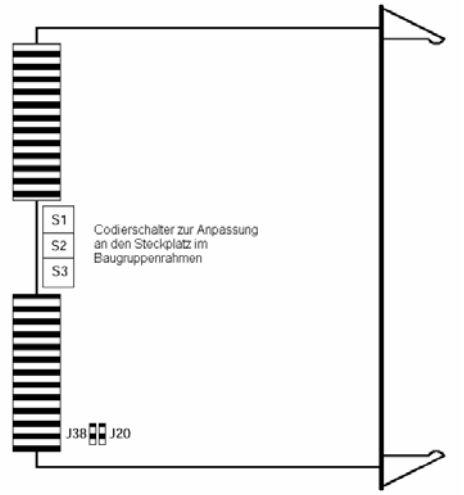


Fig. 3-19: Position of the coding switches on the CP5100 (side view)

3.6.3 Configuring steps

Only the configuring steps for one station are shown in the following. All of the other stations can be essentially handled in the same way.

3.6.3.1 Configuring in HWConfig

Parameterizing the CP5100

The CP5100 is parameterized via the appropriate tab in the object property dialog box.

To parameterize the Ethernet interface, the following relevant settings must be specified in the sub-tab “Properties of the CP5100 Ethernet interface”:

- **IP address ("Parameter" tab)**
  IP address of the module in the "dot" notation, in this case:
  141.20.135.197
- **Sub-network mask ("Parameter" tab)**
  Sub-network mask to designate the network segment,
  in this case: **255.255.0.0** for a Class B network
- **IP address of the default router ("Parameter" tab)**,
  in this case: **"Use no router"**

NOTE

It must be ensured that the IP address and sub-network mask are harmonized. For example, for a Class C IP address (193.x.y.z), the Class B sub-network mask 255.255.0.0 is not permissible. Detailed information on how to select the IP address and its value ranges is provided in the online help for CP5100.

The following diagram shows the settings, specified above under the "Parameter" tab:

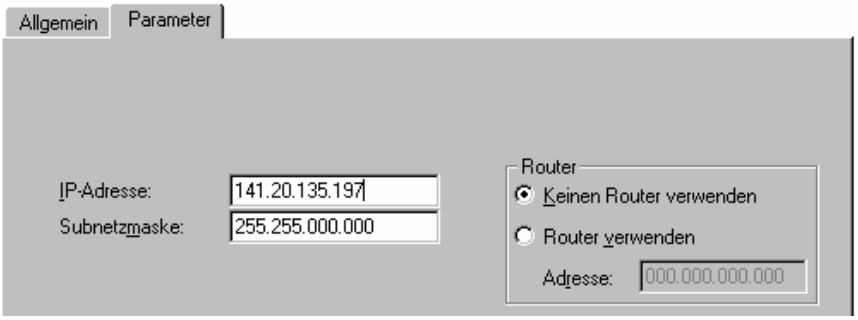


Fig. 3-20: CP5100 network settings

3.6.3.2 Configuring with CFC

Only the most important I/O are handled in the following. Deviations, which are obtained when configuring a UDP coupling, are described at the appropriate locations.

FBs required

The following function blocks are required for this example:

- **@TCPIP** central block to initialize and monitor the CP5100 module **(this is always required)**
- **CRV** receive block (in this case, optional, for the process data coupling)
- **CTV** send block (in this case, optional, for the process data coupling)

3.6.3.2.1 Central block @TCPIP

Configuring

Connection	Connection assignment (significance)
CTS	**D1800C** (module name of the configured CP5100)

CFC chart

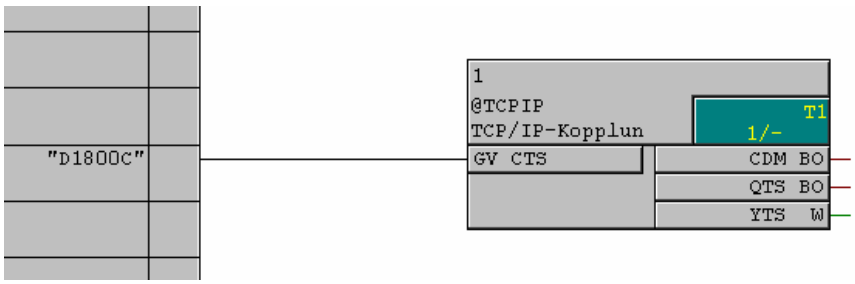


Fig. 3-21 : Connection assignment of the @TCPIP

3.6.3.2.2 Receive block CRV

"AR" connection

Initialization connection to enter an address. **Address stage 1** must be specified in addition to the channel name.

Rules for address stage 1:

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

Only ports 1024 to 65535 should be used as port number. This should be harmonized with the system administrator. Port numbers up to and including 1023 are generally reserved for “well known services” and “unix-specific services”.

Configuring

Connection	Connection assignment (significance)
CTS	**D1800C** (module name of the configured CP5100)
AR	**RXKAN1.T-01024** (address parameter, receive)
MOD	**H** (receive mode)
EN	**1** (enable)

CFC chart

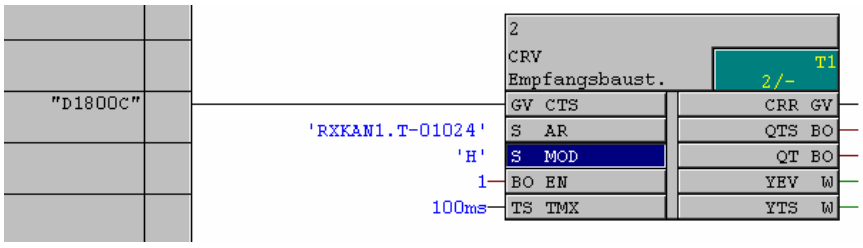


Fig. 3-22: Connection assignment of the CRV

3.6.3.2.3 Send block CTV

"AR" connection

Initialization I/O for address data. **Address stage 1** and **address stage 2** must be specified here, in addition to the channel names.

The two address stages are separated in the notation by a "**.**".

Rules for address stage 1:

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

Only ports from 1024 to 65535 should be used as port number. Port numbers up to and including 1023 are generally reserved for “well known services” and “unix-specific services”.

Rules for address stage 2:

- The first **12 digits** define the IP address of the remote coupling partner. This is entered in the so-called “dot" notation; however without specifying the separating point. Leading zeros should be specified (in this case: "**141020135198**" for IP address 141.20.135.198).
- The 13th character must be a “**-**”.
- The port number of the remote coupling partner is defined by the next **5 digits**; leading zeros should be specified (in this case: "**01024**" for port number 1024).

Configuring

Connection	Connection assignment (significance)
CTS	**D1800C** (module name of the configured CP5100)
AR	**TXKAN1.T-01024.141020135198-01024** (address parameter, send)
MOD	**H** (send mode)
EN	**1** (enable)

CFC chart

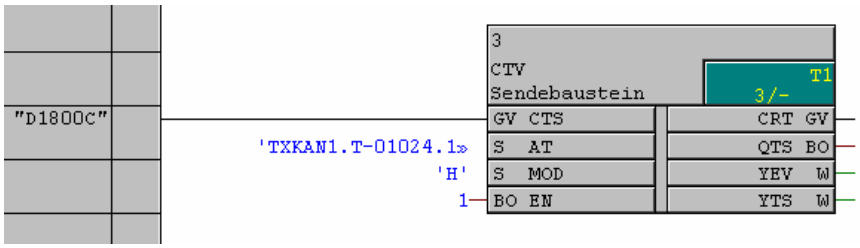


Fig. 3-23 Connection assignment of the CTV

3.6.4 Application information

3.6.4.1 Channel number

A maximum of **256 channels** can be set-up on the TCP/IP module using send and receive blocks (e.g. CTV and CRV).

The actual possible number of channels depends on the size and the number of net data and the access mechanism (handshake, refresh). 254 Kbytes of RAM on the CP5100 are available for the data interface.

Calculation

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

- **per refresh/multiple channel:** 150 bytes + 2 * number of net data bytes (size of the virtual connection)
- **per handshake/select channel:** 150 bytes + 1 * number of net data bytes

3.6.4.2 Telegram length

The telegram length is restricted as follows:

- To 55759 bytes for receiving
- 65535 bytes for sending

3.6.4.3 "Ping" on CP5100

Communications with the CP5100 can be checked using a "Ping". The module only responds to a "Ping" if the communications partner is located within the particular network or a default router (within the particular network) can establish the coupling.

3.6.4.4 Performance

The performance of the TCP/IP coupling depends on the configuration used. The following data indicate the performance values for a typical configuration.

Prerequisites

- For telegram lengths 200 and 1000 bytes, 32 send and 32 receive UDP connections are configured for each CPU (refresh).
- For a telegram length of 4000 bytes, 8 send and 8 receive connections are configured.

Results

The number of send/receive tasks per CP5100 is approx. **1600**.

The maximum data transfer rate is approx. 1.2 Mbyte/s (telegram length of 1000 bytes); this cannot be increased, in spite of the higher channel lengths.

Data transfer rate [kbytes/s]	Tasks/s	Send/receive cycle [ms] / connections	Telegram length [bytes]
320.000	1600	64 / 40	200
1142.857	1143	64 / 56	1000
1000.000	250	64 / 16	4000

3.7 PROFIBUS DP coupling (CP50M1)

3.7.1 General basics

Characteristics The CP50M1 has the following characteristics on PROFIBUS DP:

- **Master / Slave**
  Each of the two interfaces of the CP50M1 can be operated both as master (alone) or with other masters in the multi-master mode) as well as slave on PROFIBUS DP. This can be independently realized for every interface.
- **Shared input**
  Each slave connected to PROFIBUS DP is assigned just one master (the parameterizing master) and at first can only communicate with this master. Additional masters can read the slave input data using the ”Shared input”. The interfaces of the CP50M1 support this functionality as master.
- **Routing**

  CP50M1 from FW version V1.1 onwards and with D7-SYS from V7.0 onwards supports the routing function.
- **SYNC and FREEZE**
  The outputs/inputs of several slaves can be read/written in synchronism using the SYNC and FREEZE utilities. SIMATIC TDC supports these utilities as master.
- **Equidistance**
  Equidistance ia a property of PROFIBUS DP and guarantees bus cycles that are always precisely the same length.
- **Peer-to-peer data transfer**
  The configured slaves can “directly” exchange data with one another without any configuring in the CP50M1.
- **Data lengths**
  A maximum of 244 bytes can be transferred in each direction and for each slave.
- **Consistency**
  Data within a telegram is always consistent.

NOTE A maximum of six CP50M1 can be operated in parallel in one subrack.

3.7.2 Configuring

3.7.2.1 Configuring the DP system on CP50M1

For the CP50M1, the DP system is configured the same way as for SIMATIC with HW Config and the network configuring. This means that it is configured exactly the same as other DP masters (e.g. CPU 315-2DP).

The procedure is precisely desribed in the manual “Configuring Hardware and Communication Connections STEP 7“ in Chapters 3, “Configuring the Distributed I/O (DP)” and 11, “Networking Stations”.

This is the reason that only the special features and issues of the CP50M1 are discussed in the following.

3.7.2.2 Configuring communications in CFC

Function blocks

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

- A central coupling block @PRODP
- A maximum of one transmitter- and receiver function block per slave station
- Maximum of one synchronizing function block SYNPRO can be configured
- A maximum of one diagnostics function block DPDIAG and one diagnostics function block per Slave can be configured

Communications utility

The following communication utilities are permitted:

- Process data
- Parameter processing of variable-speed drives

Data transfer mode

Permitted data transfer 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.

Entries at address connection AT, AR

Special features when making data entries at address connection AT, AR when using PROFIBUS DP:

Input sequence:
"Channelname.Adressstage1.Addressstage 2"

- **Channel name**
  - max. 8 characters

- ASCII characters except "Point" and @
- channel names of all transmit- and receive blocks, which access the interfaces X1 and X2 of the CP50M1 must be different (exception for the "Multiple" data transfer mode).
- the channel name has no special significance for PROFIBUS DP.

- Input "." after the channel name
- **Address stage 1:**
  - the slave PROFIBUS address is specified as address stage 1.
  - the slave PROFIBUS address may only assigned once for each transmit- and receive channel.
  - value range: 0, 3 - 123
  - 0: means that this channel itself is used as slave channel and can be addressed from another master.
  - 3...123: addressing external slaves.
- Enter "." after address stage 1
- **Address stage 2:**
  - Comprises a maximum of 2 characters, whereby the second character is of no significance for the CP50M1.
  - **1st character:** Byte order

    "1": Standard PROFIBUS setting
    The data are transferred in the ”Motorola format” (most significant byte before the least significant byte).

    ”0”: Exception setting
    The data are transferred in the ”Intel format” (least significant byte before the most significant byte). This setting can be used for communication partners whose internal data administration uses the Intel format (e. g. SIMATIC TDC).
  -

Examples for entries at the address connection

- AT- 'Setpoint.25.1'
  - the channel with the name **setpoint** transmits data to a **slave** with the PROFIBUS address **25**.
- AR- 'RECEIVE.117.0'
  - the channel with the **RECEIVE** name receives data from a **slave** with PROFIBUS address **117**. As an exception, data are transferred in the **Intel format**.

3.7.2.3 Configuring as Slave

The procedure when configuring as slave is described in detail in „CP50M1 as PROFIBUS DP Slave“ using an example.

3.7.2.4 Shared Input

The procedure when configuring is described in detail in „Direct Data Exchange DX“ using an example.

Configuration Notes

Compared with the old CP50M0, it is no longer possible with the CP50M1 to read all the slaves.

With the CP50M1 only those slaves can be read which support the function "Direct Data Exchange". You can see which of the slaves does this from the Hardware Configuration.

If a CP50M0 is being used as parameterization master, it is not possible to parameterize a CP50M1 as reading slave.

3.7.3 Equidistance

Introduction

Equidistance on PROFIBUS DP is configured for the CP50M1 just the same as for a SIMATIC CPU (also refer to the Manual “Configuring Hardware and Communication Connections STEP 7", Chapter 3.12 "Setting Constant Bus Cycle Times for PROFIBUS Subnets").

3.7.4 SYNC/FREEZE commands

General

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

Consistency

The configuring engineer is responsible in guaranteeing that data is consistent. For the SYNC/FREEZE command, this involves consistency of data via all of the slaves involved. It goes without saying that the consistency of the input or output data of a slave is always guaranteed.

SYNC

After initiating a SYNC command, the DP master (CP50M1) waits for one DP bus circulating time, so that all of the slaves have received the new output values. The DP master then sends a SYNC broadcast telegram to the configured slave group. All slaves of this group then simultaneously update their buffered outputs.

The outputs are only again cyclically updated if the DP master sends the control command UNSYNC (EN=0 at block SYNPRO).

Ensuring data consistency:
When configuring, it must be ensured that during a DP bus circulating time, after a SYNC command has been initiated, that the SIMATIC TDC CPUs do not change the data.

FREEZE

After initiating a FREEZE command, the DP master immediately transmits a FREEZE broadcast telegram to the configured slave group. All of the slaves of this group then simultaneously read their inputs and buffer them. This input data is then available for the SIMATIC TDC CPUs after a DP bus circulating time has expired.

Input data are only again cyclically sent from the DP slave to the DP master if the DP master sends the control command UNFREEZE (EN=0 at block SYNPRO).

Ensuring data consistency:
By suitably configuring, it should be ensured that during a DP bus circulating time, after the FREEZE command has been initiated, that the input data are not evaluated by the DP master.

3.7.4.1 Configuring versions of SYNC/FREEZE

General

The terminology involved with securing data consistency are explained and various configuring versions of SYNC/FREEZE are illustrated.

Terminology

- **Bus circulating time**
  Cycle, in which the DP master (CP50M1) addresses all of the slaves once. In multi-master systems, all of the masters poll their slaves. The bus circulating time is configured using STEP 7using the baud rate, number and type of the slaves, and is computed by STEP 7.

- **Sampling time**
  This is the 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 using CFC.

NOTE Bus circulating time and sampling time are independent of one another.

- **Synccycle**
  Synccycle is a multiple integer of the sampling time. It can be configured at input CNX of function block SYNPRO.
  (Synccycle=CNX x sampling time).
  A Synccycle always starts with a sampling time. A synchronizing command is always initiated by the SYNPRO function block in the system mode at the start of a sampling time.

Configuring version 1

Configuring version 1 corresponds to most of the applications:

- Generating SYNC commands.
- The data consistency over all slaves is guaranteed.
- The Synccycle is at least twice as long as the sampling time (CNX>1).
  - the length of the transmit telegrams (outputs) for each slave may not be greater than 32 bytes.
  - all transmit blocks and the SYNPRO function block must be configured in the same sampling time.
  - the SYNPRO function block must be configured before all of the 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 circulating time must be shorter than the Synccycle minus 1 x sampling time. When operational, it should be checked as to whether the SOK output goes to “1“ once in each Synccycle, otherwise the Synccycle should be increased.

Example:

- Synccycle=3 x sampling time
- Bus circulating time=2 x sampling time
- Assumption: The SYNPRO function block calculates at the center of the sampling time (before all transmit blocks)

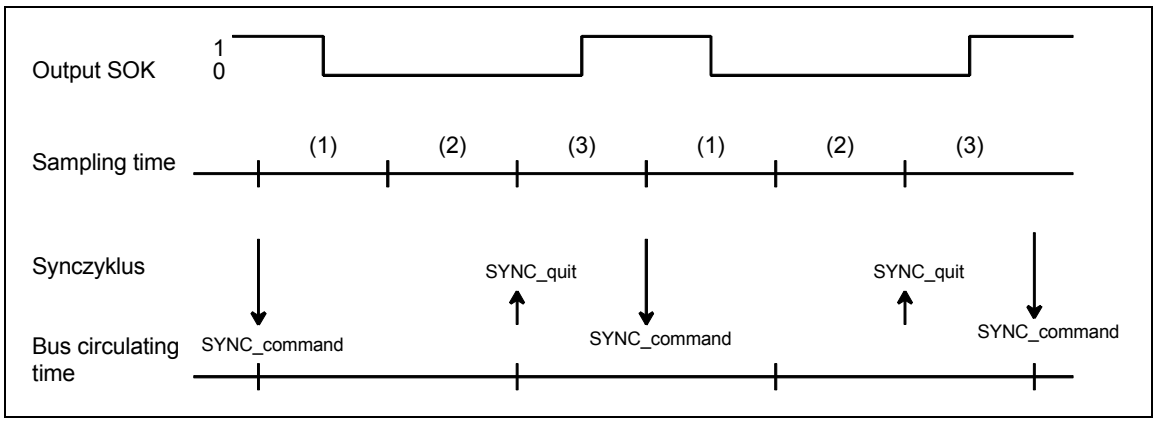


Fig. 3-24 Timing diagram, SYNC version 1

When initiating the SYNC command, the transmit blocks are inhibited (SOK=0) for two sampling times (one bus circulating time). The transmit blocks are enabled in the third sampling time after initiating the SYNC command (SOK=1).

Configuring version 2

Configuring version 2 has the highest SYNC performance:

- Generating SYNC commands.
- The data consistency over all slaves is guaranteed.
- Synccycle=sampling time (CNX=1)
  - the length of the transmit telegrams (outputs) for each slave may not be greater than 32 bytes.
  - all transmit blocks and the SYNPRO function block must be configured in the same sampling time.
  - high baud rate (>1.5 Mbaud). For lower baud rates, the time conditions can hardly be maintained.
  - the bus circulating time may only be a maximum of 50 % of the sampling time.

– the bus circulating time must also be so low, that one sampling time expires from the start up to the calculation of the function block SYNPRO. This cannot be guaranteed, but must be checked when the system is operational.

Example:

- Synccycle=sampling time
- Bus circulating time=0,3 x sampling time
- Assumption: The SYNPRO function block calculates at the center of the sampling time (before all transmit blocks)

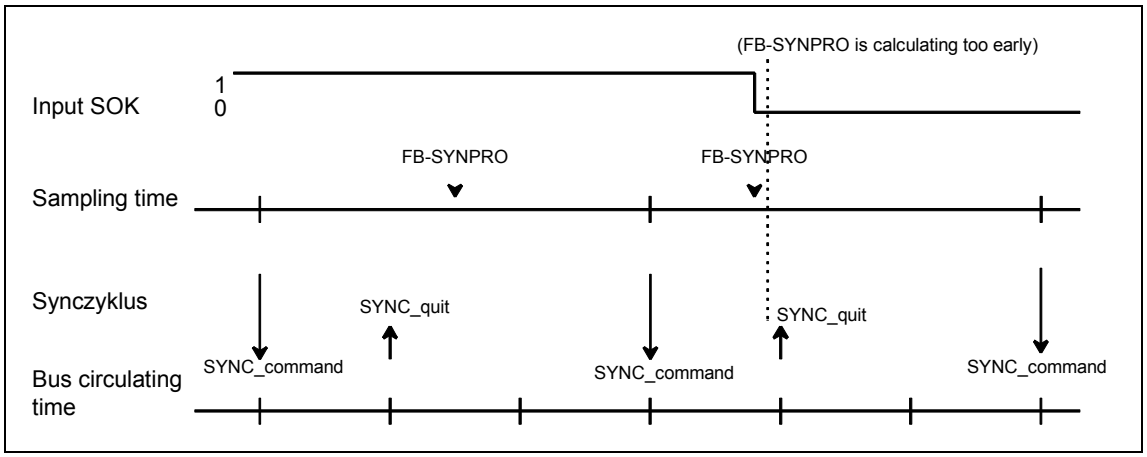


Fig. 3-25 Timing diagram, SYNC version 2

Normally, the transmit blocks are always enabled (SOK=1). If, due to time fluctuations, the SYNPRO function block is calculated before SYNC has expired (to the right in the diagram), the transmit data are not updated, but the values from the previous sampling time are transferred. The Synccycle and the data consistency are not influenced.

Instructions to achieve good SYNC functionality:

In addition to a low Synccycle, it is also necessary to have the lowest amount of jitter (time-based fluctuations) in the Synccycle. The following measures support this:

- Irregular data transfer along the DP bus should be prevented: Single-master operation; stations must not be temporarily switched-in.
- Alarm tasks should not be configured on the same SIMATIC TDC CPU. Sampling time overruns are not permissible; this would result in a SYNC command failure or a shift by a complete sampling time.
- Configure a high baud rate and short telegram lengths (the time to poll a slave is included in the jitter.).

- Configure the SYNPRO function block and all associated transmit blocks in T1=T0 (basic sampling time). The SYNC command is always initiated with the basic clock cycle interrupt. It is received with more accuracy (timing accuracy) as an interrupt, initiated in the system mode.

Configuring version 3

Configuring version 3 is for generally less frequently used applications of FREEZE:

- Generating SYNC and FREEZE or only FREEZE commands.
- The data consistency over all slaves is guaranteed.
- The Synccycle is at least 300 % longer than the sampling time (CNX>1).
  - the length of the transmit- or receive telegram (inputs or outputs) may not exceed 32 bytes per slave.
  - all transmit- and receive blocks and the SYNPRO function blocks must be configured in the same sampling time (on the same CPU).
  - the SYNPRO function block is configured as the last function block in the processing sequence.
  - output SOK of function block SYNPRO should be connected with the enable inputs of all (belonging to the slave group) transmit- and receive blocks.
- The bus circulating time must be less than the Synccycle minus 2 x the sampling time. When the system is operational, it should be checked whether the SOK output goes to “1“ once per Synccycle; otherwise the Synccycle should be increased.

Example:

- Synccycle=4 x sampling time
- Bus circulating time=2 x sampling time
- Assumption:
  The SYNPRO function block calculates at the center of the sampling time (after all of the receive- and transmit blocks)

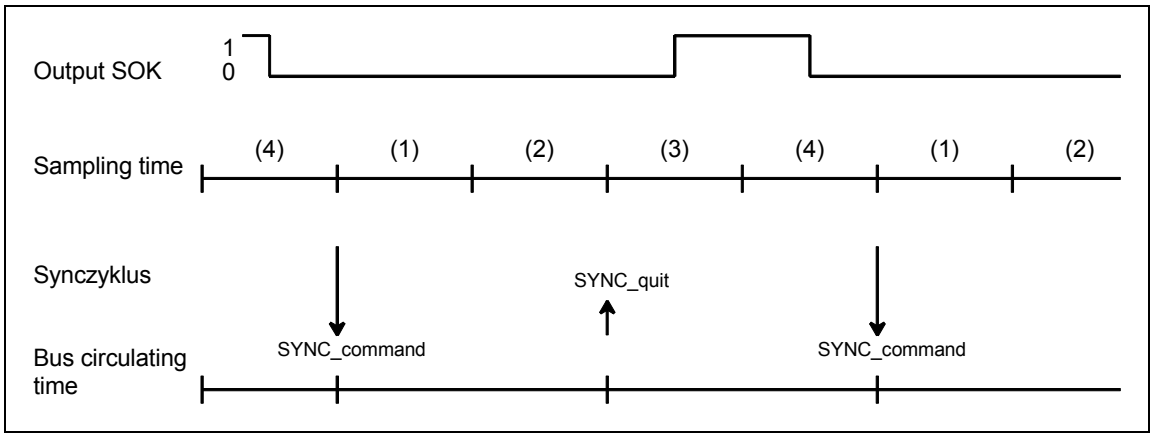


Fig. 3-26 Timing diagram SYNC version 3

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

3.7.5 Commissioning/ diagnostics

3.7.5.1 Diagnostics function block

General

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

Further information
on the diagnostic data, refer to the User Documentation for the individual slaves.

Overview, diagnostic data

- Function block **DPDIAG:** diagnostic overview The system diagnostics provides an overview as to which slave has provided diagnostic data.
- The 4 words are bit-coded.
- Each bit is assigned a slave with its PROFIBUS address corresponding to the following table.
- If the bit for the associated slave is set, then the slave has provided diagnostics data.

Output	Bit 16	Bit 15	Bit 14	...	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
D01	15	14	13	...	4	3	(2)	(1)	(0)
D02	31	30	29		20	19	18	17	16
...	...								...
D07	111	110	109		100	99	98	97	96
D08	-	-	(125)	...	116	115	114	113	112

Table 3-13 Assigning system diagnostics/data transfer list to the slave PROFIBUS address

- The data transfer list provides an overview of the slaves which were involved with data transfer within a configured time.
- The 4 words (DL1 – DL4) are bit-coded as for the system diagnostics.
- If the bit for the assigned slave is set, then data is being transferred.

Master status:

Outputs information specific to the master:

Output	Significance
MST	Status of the DP master: Stop (40h), Clear (80h), Operate (C0h)
ID	Ident-Nr. : 815Eh für CP50M1

Table 3-14 Information specific to the master

Function block **DPSLDG:** Slave diagnostics

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

Further information
on slave-specific diagnostics data, refer to the user documentation for the individual PROFIBUS slaves.

Diagnostics data of slaves

Anschluss		
ST1	Status 1	Diagnostics according to the standard 6 bytes
ST2	Status 2	
ST3	Status 3	
MPA	Master PROFIBUS address	
ID	Identification number	
D01 – D59	device-related diagnostics (refer to the User Documentation of the particular PROFIBUS slave)	

Table 3-15 Overview of the structure of the diagnostics data for Siemens DP slaves

Bits from status 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 from another master	S: Last parameter telegram was erroneous	M: Erroneous slave response	S: Requested function is not supported	S: Diagnostics entry in the specific diagnostics area	S: Config. data dont match	S: Slave still not ready for data transfer	M: Slave cannot be addressed on the bus
Status 2 (ST2)	M: Slave entered as "not active"	(not used)	S: Slave has received a Sync command	S: Slave has received the Freeze command	S: Response monitoring activated	S: 1 (fixed)	S: Diagnostic data must be retrieved	S: Parameterization and configuring required
Status 3 (ST3)	S/M: Not all diagnostics data can be transferred	-	-	-	-	-	-	-

Table 3-16 Significance of the individual bits, status 1, 2 and 3

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

Master PROFIBUS address

- PROFIBUS address of the master which had parameterized this slave.

If this slave is not parameterized, then FFh is used.

Identification number (ID)

- This identifies the slave type.

All additional diagnostic data are slave-specific.

Generally (standard DP slave) the diagnostic blocks follow: 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. The diagnostics block is identified by bit 7 and bit 8:

Bit 7, 8 of 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

Table 3-17 Significance of bit 7 and bit 8 of the header byte

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.

Function block DIAPRO (refer to chapter 3.7.1.5)

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

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

- **Error class>0:** An error is present. Function block @PRODP issues a communications error (flashing ”LED” on the CP50M1 module). For users, these connections have hardly any significance as the corresponding communication errors can be read-out from the diagnostics buffer. When required, you will be asked about the values at these connections if you contact the hotline with critical faults.

3.8 PROFIBUS DP coupling (CP50M0)

Additionally required hardware and software

The following hardware and software are additionally required to configure and run the PROFIBUS DP coupling:

- COM PROFIBUS
  Order No. 6ES5 895-6SE03
- SS52load
  SS52load is included in COM PROFIBUS from V3.1.
- DP-capable PC card to download the COM database via COM PROFIBUS:

Characteristics

SIMATIC TDC has the following characteristics on PROFIBUS DP:

- **Master**
  The CP50M0 communications module can be operated on PROFIBUS both alone (stand alone) and with other masters in multi-master operation.
- **Slave**
  In addition to the master functionality, there is also the slave functionality. Both of these functionalities can be used simultaneously or separately.
- **Shared input**
  Each slave connected to PROFIBUS DP is assigned just one master (the parameterizing master) and at first can only communicate with this master. Additional masters can read the slave input data using the ”Shared input”. SIMATIC TDC supports this functionality as master and slave.
- **SYNC and FREEZE**
  The outputs/inputs of several slaves can be read/written in synchronism using the SYNC and FREEZE utilities. SIMATIC TDC supports these utilities as master.
- **Data lengths**
  A maximum of 244 bytes can be transferred in each direction and for each slave.
- **Data transfer times**
  For short telegrams (up to 32 byte), only the SIMADYN D sampling time and the DP bus circulating time are included in the data transfer time. For longer telegrams, the software processing times of the SS52 communications module must also be included (max. 5 ms).
- **Consistency**
  Data within a telegram is always consistent.

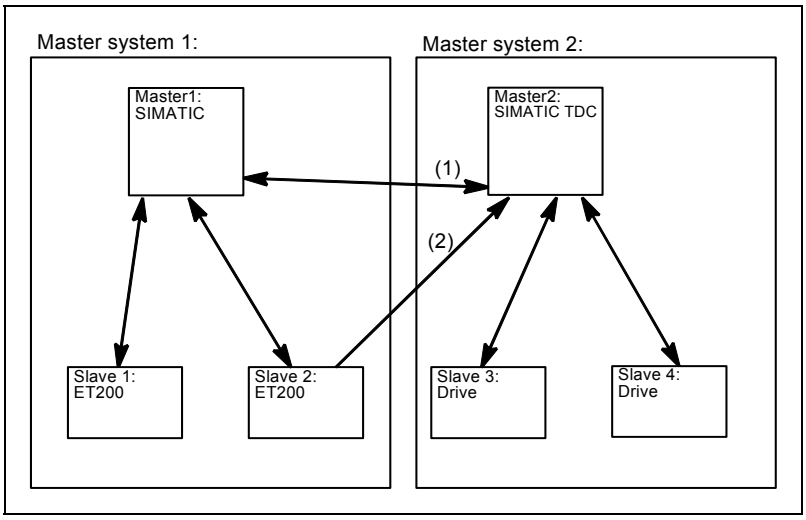


Fig. 3-27 Multi-master system with slave functionality (1) and shared input (2)

3.8.1 Configuring with D7-SYS

Function blocks

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

- A central coupling block @PRODP
- A maximum of one transmitter- and receiver function block per slave station
- Maximum of one synchronizing function block SYNPRO can be configured
- A maximum of one diagnostics function block DIAL2A can be configured

Communications utility

The following communication utilities are permitted:

- Process data
- Parameter processing of variable-speed drives

Data transfer mode

Permitted data transfer mode:

- Refresh
- For receivers, optionally also multiple

3.8.1.1 Central coupling block

Baud rate and PROFIBUS address

The baud rate and PROFIBUS address are specified, on one hand by CFC (function block @PRODP) and on the other hand by COM PROFIBUS.
The following must be observed regarding the validity of these two parameters:

- If a COM database has still not been loaded, then
  - the parameters specified by CFC are valid.
  - the CP50M0 communications module waits for a COM database to be downloaded.
- If a COM database is loaded and the baud rate and PROFIBUS address are the same as those configured with the CFC, then
  - the COM database is activated.
  - communications module CP50M0 starts with net data transfer.
- If a COM database is loaded, but the baud rate or PROFIBUS address does not coincide with that configured in CFC, then
  - the parameters specified by CFC are valid.
  - the module waits for download. (the existing COM database can also be activated, by correcting the baud rate and PROFIBUS address at the central block of the COM configuring.)

3.8.1.2 Address connections AT, AR

Entries at address connection AT, AR

Special features when making data entries at address connection AT, AR when using PROFIBUS DP:

Input sequence:
"Channelname.Adressstage1.Addressstage 2"

- **Channel name**
  - max. 8 characters
  - ASCII characters except "Point" and @
  - channel names of all transmit- and receive blocks, which access the same SS52 communications module must be different (exception for the "Multiple" data transfer mode).
  - the channel name has no special significance for PROFIBUS DP.
- Input "." after the channel name
- **Address stage 1:**
  - the slave PROFIBUS address is specified as address stage 1.
  - the slave PROFIBUS address may only assigned once for each transmit- and receive channel.
  - value range: 0, 3 - 123
  - 0: means that this channel itself is used as slave channel and can be addressed from another master.
  - 3...123: addressing external slaves.
- Enter "." after address stage 1

- **Address stage 2:**
  - consists of a maximum of 2 characters.
  - **1st character:** Byte order

    "1": Standard PROFIBUS setting
    The data are transferred in the ”Motorola format” (most significant byte before the least significant byte).

    ”0”: Exception setting
    The data are transferred in the ”Intel format” (least significant byte before the most significant byte). This setting can be used for communication partners whose internal data administration uses the Intel format (e. g. SIMATIC TDC).
  - **2nd character:** Optional, only receiver
    "R":
    The access is realized as second master which reads data. ”R” can only be entered for receive channels. (”Shared input”)

    If a 2nd character is not specified, then the slave can be accessed as parameterizing master.

Examples for entries at the address connection

- AT- 'Setpoint.25.1'
  - the channel with the name **setpoint** transmits data to a **slave** with the PROFIBUS address **25**.
- AR- 'RECEIVE.117.0'
  - the channel with the **RECEIVE** name receives data from a **slave** with PROFIBUS address **117**. As an exception, data are transferred in the **Intel format**.
- AR- 'Input.33.1R'
  - the channel with the name **input** receives data from a **slave** with PROFIBUS address **33** as **(second) master which reads data**.
- AT- 'Slavelst.0.1'
  - the channel with the name **slavelst** transmits data as **slave** to a **DP master**.

3.8.1.3 SYNC/FREEZE commands

General

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

Consistency

The configuring engineer is responsible in guaranteeing that data is consistent. For the SYNC/FREEZE command, this involves consistency of data via all of the slaves involved. It goes without saying that the consistency of the input or output data of a slave is always guaranteed.

SYNC

After initiating a SYNC command, the DP master (CP50M0) waits for one DP bus circulating time, so that all of the slaves have received the new output values. The DP master then sends a SYNC broadcast telegram to the configured slave group. All slaves of this group then simultaneously update their buffered outputs.

FREEZE

Ensuring data consistency:
When configuring, it must be ensured that during a DP bus circulating time, after a SYNC command has been initiated, that the SIMATIC TDC CPUs do not change the data.

After initiating a FREEZE command, the DP master immediately transmits a FREEZE broadcast telegram to the configured slave group. All of the slaves of this group then simultaneously read their inputs and buffer them. This input data is then available for the SIMATIC TDC CPUs after a DP bus circulating time has expired.

Ensuring data consistency:
By suitably configuring, it should be ensured that during a DP bus circulating time, after the FREEZE command has been initiated, that the input data are not evaluated by the DP master.

3.8.1.4 Configuring versions of SYNC/FREEZE

General

The terminology involved with securing data consistency are explained and various configuring versions of SYNC/FREEZE are illustrated.

Terminology

- **Bus circulating time**
  Cycle, in which the DP master (CP50M0) addresses all of the slaves once. In multi-master systems, all of the masters poll their slaves. The bus circulating time is configured using COM PROFIBUS using the baud rate, number and type of the slaves, and is computed by COM PROFIBUS. It can be read-out there using the menu command **Bus parameters**, as ”Typical data cycle time”.

- **Sampling time**
  This is the 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 using CFC.

NOTE Bus circulating time and sampling time are independent of one another.

- **Synccycle**
  Synccycle is a multiple integer of the sampling time. It can be configured at input CNX of function block SYNPRO.
  (Synccycle=CNX x sampling time).
  A Synccycle always starts with a sampling time. A synchronizing command is always initiated by the SYNPRO function block in the system mode at the start of a sampling time.

Configuring version 1

Configuring version 1 corresponds to most of the applications:

- Generating SYNC commands.
- The data consistency over all slaves is guaranteed.
- The Synccycle is at least twice as long as the sampling time (CNX>1).

- the length of the transmit telegrams (outputs) for each slave may not be greater than 32 bytes.
- all transmit blocks and the SYNPRO function block must be configured in the same sampling time.
- the SYNPRO function block must be configured before all of the 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 circulating time must be shorter than the Synccycle minus 1 x sampling time. When operational, it should be checked as to whether the SOK output goes to “1“ once in each Synccycle, otherwise the Synccycle should be increased.

Example:

- Synccycle=3 x sampling time
- Bus circulating time=2 x sampling time
- Assumption: The SYNPRO function block calculates at the center of the sampling time (before all transmit blocks)

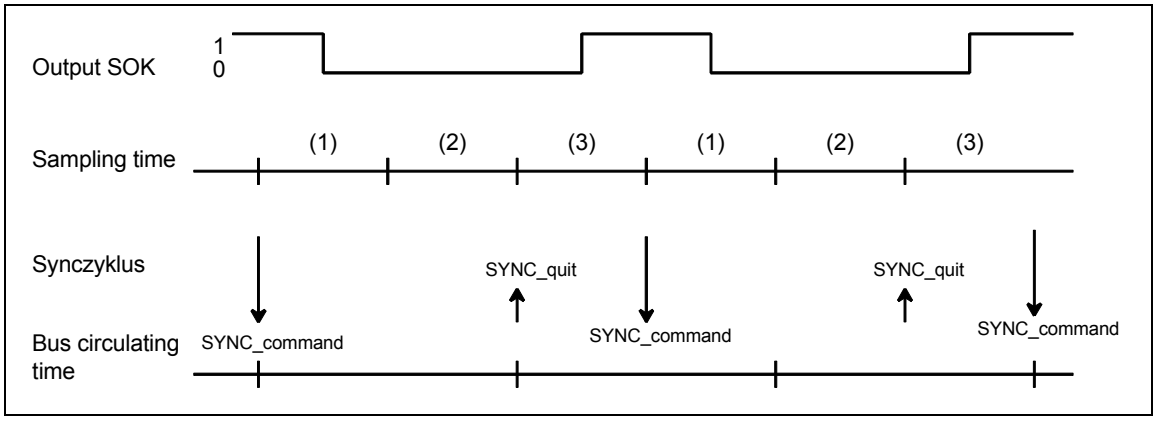


Fig. 3-28 Timing diagram, SYNC version 1

When initiating the SYNC command, the transmit blocks are inhibited (SOK=0) for two sampling times (one bus circulating time). The transmit blocks are enabled in the third sampling time after initiating the SYNC command (SOK=1).

Configuring version 2

Configuring version 2 has the highest SYNC performance:

- Generating SYNC commands.
- The data consistency over all slaves is guaranteed.
- Synccycle=sampling time (CNX=1)
  - the length of the transmit telegrams (outputs) for each slave may not be greater than 32 bytes.

- all transmit blocks and the SYNPRO function block must be configured in the same sampling time.
- high baud rate (>1.5 Mbaud). For lower baud rates, the time conditions can hardly be maintained.
- the bus circulating time may only be a maximum of 50 % of the sampling time.
- the bus circulating time must also be so low, that one sampling time expires from the start up to the calculation of the function block SYNPRO. This cannot be guaranteed, but must be checked when the system is operational.

Example:

- Synccycle=sampling time
- Bus circulating time=0,3 x sampling time
- Assumption: The SYNPRO function block calculates at the center of the sampling time (before all transmit blocks)

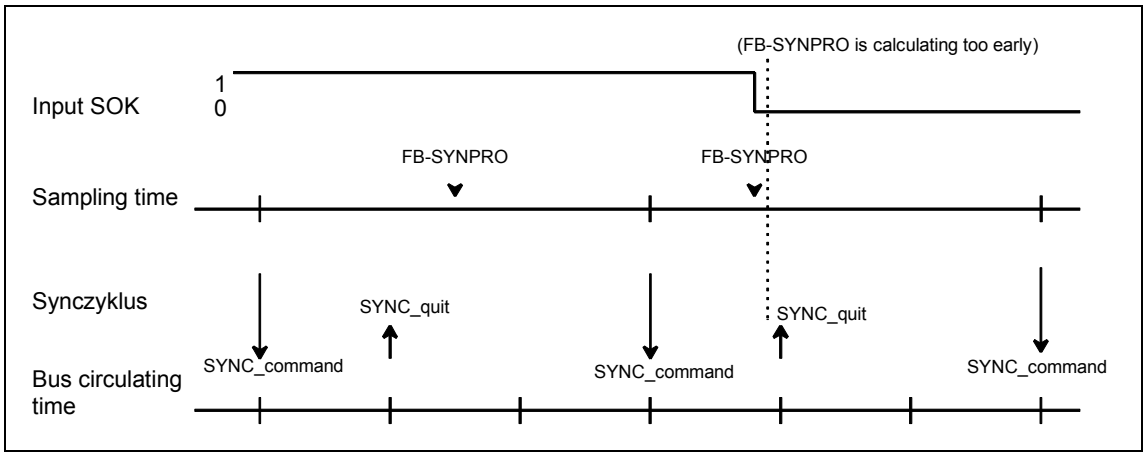


Fig. 3-29 Timing diagram, SYNC version 2

Normally, the transmit blocks are always enabled (SOK=1). If, due to time fluctuations, the SYNPRO function block is calculated before SYNC has expired (to the right in the diagram), the transmit data are not updated, but the values from the previous sampling time are transferred. The Synccycle and the data consistency are not influenced.

Instructions to achieve good SYNC functionality:

In addition to a low Synccycle, it is also necessary to have the lowest amount of jitter (time-based fluctuations) in the Synccycle. The following measures support this:

- Irregular data transfer along the DP bus should be prevented: Single-master operation; stations must not be temporarily switched-in.
- Alarm tasks should not be configured on the same SIMATIC TDC CPU. Sampling time overruns are not permissible; this would result in a SYNC command failure or a shift by a complete sampling time.

- Configure a high baud rate and short telegram lengths (the time to poll a slave is included in the jitter.).
- Configure the SYNPRO function block and all associated transmit blocks in T1=T0 (basic sampling time). The SYNC command is always initiated with the basic clock cycle interrupt. It is received with more accuracy (timing accuracy) as an interrupt, initiated in the system mode.

Configuring version 3

Configuring version 1 (3) is for generally less frequently used applications of FREEZE:

- Generating SYNC and FREEZE or only FREEZE commands.
- The data consistency over all slaves is guaranteed.
- The Synccycle is at least 300 % longer than the sampling time (CNX>1).
  - the length of the transmit- or receive telegram (inputs or outputs) may not exceed 32 bytes per slave.
  - all transmit- and receive blocks and the SYNPRO function blocks must be configured in the same sampling time (on the same CPU).
  - the SYNPRO function block is configured as the last function block in the processing sequence.
  - output SOK of function block SYNPRO should be connected with the enable inputs of all (belonging to the slave group) transmit- and receive blocks.
- The bus circulating time must be less than the Synccycle minus 2 x the sampling time. When the system is operational, it should be checked whether the SOK output goes to “1“ once per Synccycle; otherwise the Synccycle should be increased.

Example:

- Synccycle=4 x sampling time
- Bus circulating time=2 x sampling time
- Assumption:
  The SYNPRO function block calculates at the center of the sampling time (after all of the receive- and transmit blocks)

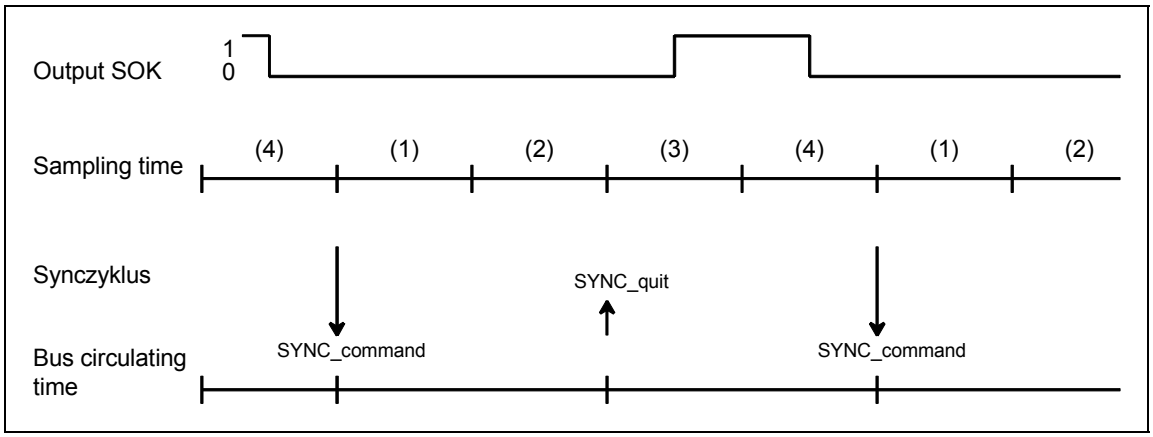


Fig. 3-30 Timing diagram SYNC version 3

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

3.8.1.5 Diagnostics function block

General

Master- or slave-specific diagnostic information can be output from PROFIBUS DP using the DIAPRO function block.

The diagnostic data to be output are selected using input SEL. It is output at D01 to D08.

Further information
on the diagnostic data, refer to the User Documentation "COM PROFIBUS" or in the User Documentation for the individual slaves.

Overview, diagnostic data

SEL=0: No diagnostic data

- The block does not output any valid diagnostic data.

SEL=126: System diagnostics

- The system diagnostics provides an overview as to which slave has provided diagnostic data.
- The 8 words are bit-coded.
- Each bit is assigned a slave with its PROFIBUS address corresponding to the following table.
- If the bit for the associated slave is set, then the slave has provided diagnostics data.

Output	Bit 16	Bit 15	Bit 14	...	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
D01	15	14	13	...	4	3	(2)	(1)	(0)
D02	31	30	29		20	19	18	17	16
...	...								...
D07	111	110	109		100	99	98	97	96
D08	-	-	(125)	...	116	115	114	113	112

Table 3-18 Assigning system diagnostics/data transfer list to the slave PROFIBUS address

SEL=127: Data transfer list

- The data transfer list provides an overview of the slaves which were involved with data transfer within a configured time (COM PROFIBUS).
- The 8 words are bit-coded as for the system diagnostics.
- If the bit for the assigned slave is set, then data is being transferred.

SEL=128: Master status

Outputs information specific to the master (for users, the low byte of D01 is relevant; the significance of the other outputs has been documented, but hasn't been explained in any more detail):

Output		Significance
D01	low byte	Status of the DP master: Stop (40h), Clear (80h), Operate (C0h)
	high byte	Ident No. SS52 (high byte)=80h
D02	low	Ident No. SS52 (low byte)=37h
	high	(irrelevant)
D03...D08		

Table 3-19 Information specific to the master

SEL=3 ... 123: Slave diagnostics

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

Further information
on slave-specific diagnostics data, refer to the user documentation "COM PROFIBUS" and the User Documentation for the individual PROFIBUS slaves.

Diagnostics data of SIEMENS DP slaves

Slave type / Connection		SPC slaves, general	ET 200U	ET 200B	ET 200K	SPM slave	ET 200C 8DE/8DA	DP stand. slaves	
D01	low	Status 1							Diagnostics according to the standard 6 bytes
	high	Status 2							
D02	low	Status 3							
	high	Master PROFIBUS address							
D03	low	Identification number, high byte							
	high	Identification number, low byte							
D04	low	Header, device-related diagnostics							Device-specific diagnostics
	high	Device diagnostics U		Device diagnos. B	0	0	0		
D05	low	Header identification-related diagnostics		0	0	0	0		
	high	BG 7-0		0	Channel 7-0				
D06	low	BG 15-8		0	Channel 15-8		0		
	high	BG 23-16		0	Channel 23-16		0		
D07	low	BG 31-24		0	Channel 31-24		0		
	high	Additional device-specific diagnostics	Irrelevant						
D08	low		Irrelevant						
	high		Irrelevant						

Table 3-20 Overview of the structure of the diagnostics data for Siemens DP slaves

Bits, status 1, 2 and 3

	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Status 1 (D01 low byte)	S: Slave was parameterized from another master	S: Last parameter telegram was erroneous	M: Erroneous slave response	S: Requested function is not supported	S: Diagnostics entry in the specific diagnostics area	S: Config. data dont match	S: Slave still not ready for data transfer	M: Slave cannot be addressed on the bus
Status 2 (D01 high byte)	M: Slave entered as ”not active”	(not used)	S: Slave has received a Sync command	S: Slave has received the Freeze command	S: Response monitoring activated	S: 1 (fixed)	S: Diagnostic data must be retrieved	S: Parameterization and configuring required
Status 3 (D02 low byte)	S/M: Not all diagnostics data can be transferred	-	-	-	-	-	-	-

Table 3-21 Significance of the individual bits, status 1, 2 and 3

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

Master PROFIBUS address

- PROFIBUS address of the master which had parameterized this slave.

If this slave is not parameterized, then FFh is used.

Identification number

- High/low byte:
  This identifies the slave type.

All additional diagnostic data are slave-specific.

Generally (standard DP slave) the diagnostic blocks follow: 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. The diagnostics block is identified by bit 7 and bit 8:

Bit 7, 8 of 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

Table 3-22 Significance of bit 7 and bit 8 of the header byte

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.

Output of additional slave diagnostics data

- Diagnostic bytes 17 to 32 of a slave are output with SEL=1002 to SEL=1123.

3.8.2 Configuring with COM PROFIBUS

General

COM PROFIBUS (Windows) should be used when configuring (it is also possible to use the earlier COM ET200 Version 2.1 for configuring). Using COM PROFIBUS you can define:

- The number and configuration of the nodes connected to the PROFIBUS DP bus system
- The baud rate
- Important parameters when using the PROFIBUS DP bus system

SIMATIC TDC -specific information on COM PROFIBUS:

- Configure the CP50M0 communications module as SIMADYN D SS52 station type ("SIMADYN" family).
- The input and output addresses should not be specified.
- After completing the configuring, the database is downloaded into CP50M0 via the DP bus using the menu command **File > Export > DP master**.
- Alternatively, it is also possible to download via RS232. The following menu command is used to start loading the SS52: **File > Export > SIMADYN Master**.

3.8.2.1 Harmonizing with data configured in CFC

Rules

The configured software should be harmonized with one another as follows:

- The baud rate and the actual PROFIBUS address must be the same.

- The slaves, configured in COM must each have, in the receive- and transmit directions, a CRV/CTV function block configured in the CFC. This is assigned via the PROFIBUS address (address stage 1 at the address connection).
- The length of the input- (receive-) and output- (transmit-) data per slave must coincide.

Error- and alarm information

The rules (syntax) are checked. Error- or alarm information is issued if these rules (syntax) are not observed:

- Communications error field (flashing "C" on the CPU module), or output YTS at function block CRV/CTV
- Output ECO at function block @PRODP

NOTE

The following rules /syntax) are not checked:
The net data structure of the communication partners must be the same.

If this is not observed, the data could be incorrectly interpreted (e. g. bytes could be interchanged within a data word) between the communication partners.

Net data structure

For SIMATIC TDC, the net data structure with CFC is specified by configuring the virtual connections (refer to the Chapter Communications utility, process data).

- For most of the PROFIBUS slaves, the net data structure is specified using COM PROFIBUS by entering identification codes in the "Configuring" window.

3.8.2.2 CP50M0 as PROFIBUS slave

Configuring

The CP50M0 communications module can be configured as pure slave or combined as master and slave:

- CP50M0 as pure slave does not require COM to be configured:
  Input SLA should be set to 1 or 2 at function block @PRODP. A function block CRV and/or CTV should then be configured next to it. The address stage 1 at connection AR/AT should be set to "0".
- CP50M0 combined as master and slave
  Input SLA at function block @PRODP should be set to "0" (default value).
  - The bus is configured using COM PROFIBUS. A database ("master system") is created for each PROFIBUS master. This is used to download the particular master.
  - If the master is configured using another tool, when configuring the CP50M0 slave, a fictitious master must be configured in COM PROFIBUS. It should be ensured that the bus parameters are correctly set: It is recommended to increase the number of active stations and the token rotation time in both configuring tools.

3.8.2.3 Loading the database

Versions

There are two ways to load the database:

Loading via PROFIBUS DP

- Loading via PROFIBUS DP is the version which is the more user friendly.
  However, certain restrictions must be observed.
- A DP-capable PC card (currently available cards can be requested from the support facility product)
- The driver for the PC card is installed together with COM PROFIBUS. Loading is realized in COM PROFIBUS using the menu command **File > Export > DP master**.

Loading via RS232

- Using the "SS52LOAD" program, a database, generated from the COM PROFIBUS is loaded as binary file into the CP50M0 module via the RS232 interface.
- SS52LOAD is integrated in COM PROFIBUS (from Version 3.1).
- Restriction:
  If the Sync function block SYNPRO is configured, then the synchronous mode must be disabled (enable input EN=0), so that the download functions.
- The binary file (*.2bf) is generated in COM PROFIBUS using the menu command **File > Export > Binary file**.
- Loading is realized with SS52LOAD with the menu command **File > Download** .
- The RS232 interface is located together with the PROFIBUS interface on the 9-pin connector to the corresponding CP50M0-interface. The customer must assemble his own cable to establish the connection to the COM port of the PC. RS232 assignment at socket (no standard):
  - 2 - TxD
  - 7 - RxD

CAUTION There is a danger of interchanging connections for the RS232 assignment.

3.8.3 Start-up/diagnostics

3.8.3.1 LEDs

Yellow LED

Contrary to the other communication modules, for the CP50M0 communications module, the yellow LED does not directly indicate the bus activity. The yellow LED provides information about the DP bus and the COM database.

Green LED

The green LED provides general information about the CP50M0 communications module and about synchronization with function block @PRODP from SIMATIC TDC.

LED	Green	Yellow
Dark	CPU not running	No bus operation (during run-up).
Flashes quickly (every 0.2 s)	Fatal error • Remedy: Read-out the error class and -code at function block @PRODP and inform Siemens AG.	Bus error (e. g. short circuit) • Remedy: Check the cable and the other bus nodes.
Flashes (every 1 s)	Wait and synchronize to the SIMATIC TDC-CPU • Remedy: Check the configuring of function block @PRODP.	COM database not available or not activated (also during download) • Remedy: Load the database.
Flashes slowly (every 2 s)	-	CFC- and COM configuring do not match 100%. Only restricted bus operation is possible • Remedy: Adapt the CFC- and COM configuring so that they match.
Steady	Communications module CP50M0 and synchronization to SIMATIC TDC CPUs OK.	Bus operation with activated COM database OK.

Table 3-23 Significance of the LEDs of the CP50M0 communication module

Behavior at run-up

- After power-on, both diodes are briefly lit and then go dark again.
- Only the green LED is lit during the run-up time (approx. 5 seconds).
- When the system is OK, the yellow LED is lit after the run-up time has expired.
- After a reset, both LEDs initially stay in the last condition until the software again controls the LEDs.

Characteristics at download

- During download, the yellow LED flashes (this is extremely short at high baud rates).
- After this, the behavior is the same as for run-up.

The LEDs do not provide information as to whether all of the slaves are available at the bus and have been correctly parameterized. If data transfer with a slave is not OK, then this is flagged at the associated function block (YEV=0x0002 or YTS=0x6014) using a ”break” ID.

Information regarding the current status of individual slaves is obtained using the diagnostics function block DIAPRO.

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

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

- **Error class=0:** An alarm is present. In some cases this alarm can be removed without a SIMATIC TDC reset. If there are several alarms, then the alarm of the lowest number is displayed.
- **Error class>0:** An error is present. Function block @PRODP issues a communications error (flashing ”C” on the CPU module). After the error has been removed, the SIMATIC TDC subrack must be reset.

Error class	Error code	Significance
0 (alarm)	0	O.K.
	1	COM database present, but not activated as the baud rate and the PROFIBUS address with connections BDR and MAA do not match. • Remedy: Harmonize the baud rate and PROFIBUS address in the CFC and COM configuring.
	2	No COM database available. • Remedy: Load the database.
	3	The COM database is presently being downloaded with subsequent start-up.
	4	The channels to DP nodes, configured with CFC, which are configured in the COM database, are missing. This status can also temporarily occur after a SIMATIC TDC run-up. The DP nodes are not addressed. • Remedy: Harmonize the CFC- and COM configuring.
	5	(not used)
	6	There is at least one channel configured with CFC which does not match the COM database. The associated SIMATIC TDC-FB has issued a communications error (flashing ”C”). • Remedy: Harmonize the CFC- and COM configuring.
	7	There is at least one channel configured with CFC, which essentially does not match the COM database. The associated SIMATIC TDC-FB has issued a communications error (flashing ”C”). • Correct the CFC configuring.
	8	Resource bottleneck. Not all of the CFC channels are processed. • Remedy: Reduce the CFC configuring (communication channels).
	9	There are two channels, which wish to transmit data to the same slave or receive data from it. The SIMATIC TDC-FB which is associated with the channel which addressed the slave later, has issued a communications error (flashing ”C”). • Remedy: Correct the CFC configuring.
	10	Bus operation temporarily faulted. • Remedy: Check the cable and bus nodes.
>1 (internal error)	(any)	• Remedy: Note the error class and error code and inform Siemens AG.

Table 3-24 Significance of the error class and error code

3.8.3.3 Application example, PROFIBUS DP coupling

General

The application example describes a typical configuration consisting of:

- SIMOVERT 6SE70
- ET200U
- ET200B
- SIMATIC S5

It is assumed that you are knowledgeable about configuring SIMATIC TDC as well as the CFC configuring language.

NOTE

Only those activities are described in detail which are of significance for this particular configuration. Versions or additional components are touched-on but not discussed in detail. In the text, these positions are identified with the symbol located on the right.

The following subjects are discussed in this application example:

- **Typical configuration**
  Description of a typical configuration for SIMATIC TDC connected to PROFIBUS DP with the associated system requirements.
- **Configuring under CFC**
  Grouping of the PROFIBUS DP specific blocks and their configuring in the typical configuration.
- **Configuring the CP50M0... communications module**
  Configuring the CP50M0 communication module using the COM PROFIBUS 3.0 parameterizing software and the download tool "SS52load".

3.8.3.4 Typical configuration and system requirements

General

The following systems and devices are selected as typical configuration, whereby the specified PROFIBUS addresses were randomly defined:

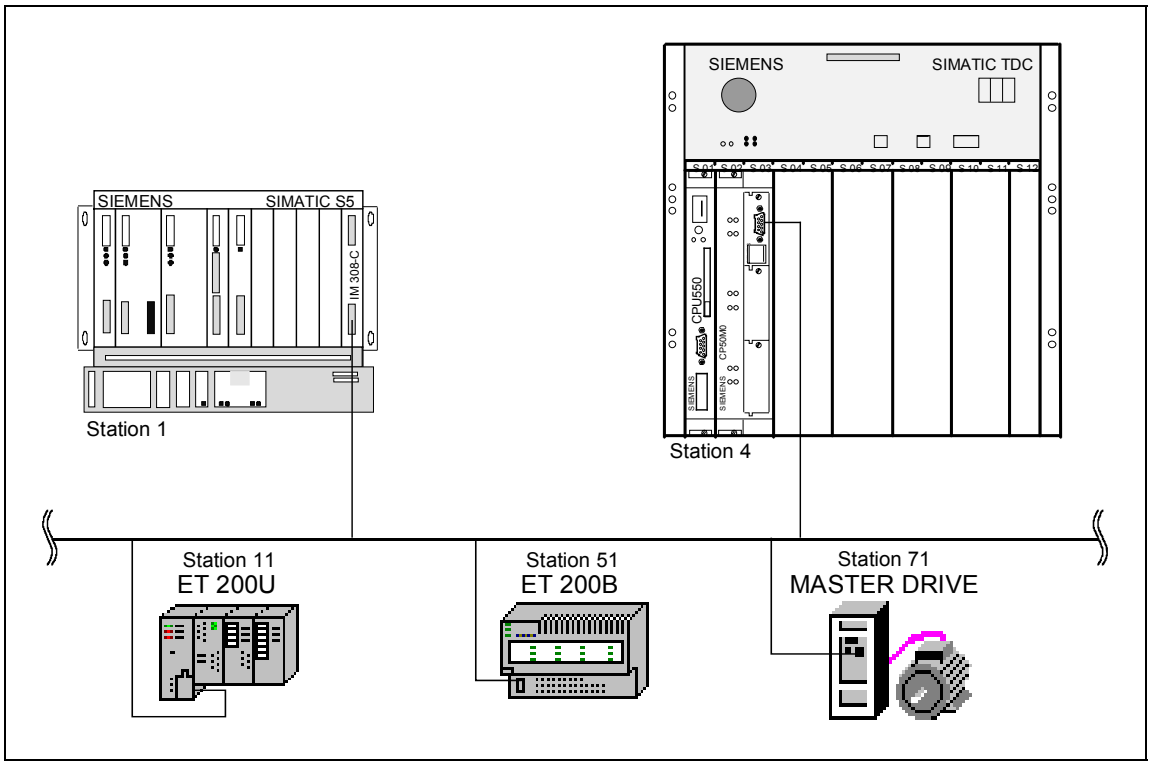


Fig. 3-31 Typical configuration

Communications partner

The SIMATIC TDC communication partners (station 4) are as follows:

- **SIMATIC S5-105U (station 1) as master to SIMATIC TDC:**
  The CP50M0 as a master (S5) which polls the SIMATIC TDC. Data transfer (quantity and amount of process data) between the two controls can be freely configured. The following was defined:
  - S5 ⇨ SIMATIC TDC: Three words (input/output), one word (input), one byte (input), one byte (input)
  - SIMATIC TDC ⇨ S5: Three words (input/output), one word (output), one byte (output)
- **SIMOVERT MASTER DRIVE with CB1 (station 71) as slave:**
  Five defined PPO types are available for data transfer with this node. PPO: Parameter process data object structure of the net data for variable-speed drives. There is net data, which either consists of parameter ID values (PKW) and process data (PZD) (PPO types 1,2,5) or only process data (PPO types 3,4).
  In this configuring example, PPO type 3 is configured. In this case, two words (control word and main setpoint) are transmitted and two words (status word and main actual value) received.

- **ET 200 B (station 51) as slave:**
  When using this slave type, a precise type must be selected which then automatically defines data transfer. For 8DI/8DO types, one byte is output and one byte is read-in.

- **ET 200 U (station 11) as slave:**
  For this ET 200 U configuration (three digital output modules and a digital input module) three bytes are output and one byte is read-in.

3.8.3.5 Check list of the required hardware and software components for SIMATIC TDC

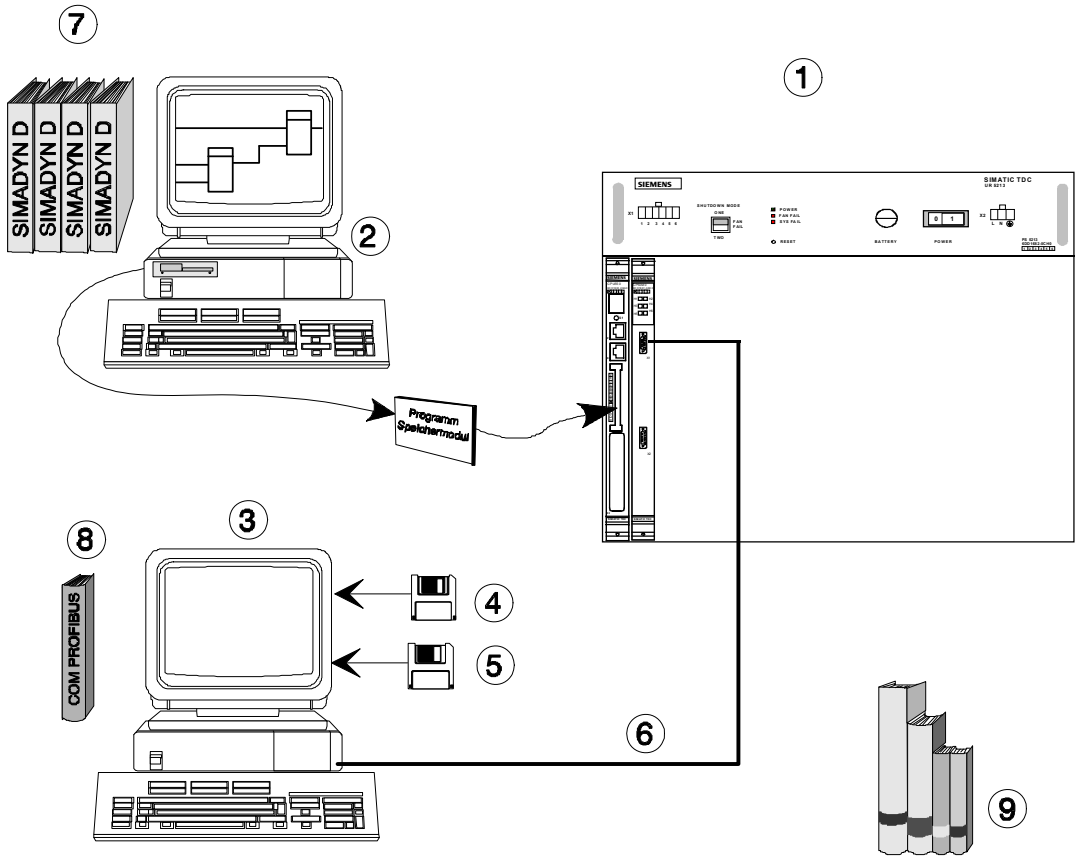


Fig. 3-32 Hardware and software components for SIMATIC TDC

Legend

1	SIMATIC TDC unit consisting of at least: Subrack, CPU, program memory sub-module, communications module CP50M0
2	CFC configuring device: PC with Windows 95/NT as operating system, STEP 7 software, option package D7-SYS and PCMCIA drive
3	PC to operate "COM PROFIBUS" and "SS52load" (this can be the same PC as for CFC), with: 3.5" floppy disk drive, one serial interface, Windows 3.1x or Windows 95 operating system
4	"COM PROFIBUS 3.0" parameterizing software: Software to generate the PROFIBUS DP bus configuration
5	Download software "SS52load": Software to transfer the DP configuration generated with "COM PROFIBUS" to CP50M0 via the COM port (RS 232) of a PC.
6	RS232 line: Connection between the CP50M0 (in every 9-pin connector of the CP50M0, in addition to the RS485 of the Profibus, there is also an RS232: 2-TxD; 7-RxD) and a PC COM port (RS232). This cable must be assembled according to the specifications (refer to Chapter Downloading the COM database onto the CP50M0) as the RS232 of the CP50M0 is not standard! If data is downloaded via the bus (RS 485), using a communications processor CP 5411 (additional plug-in card in the PC), then the "SS52load" tool and the RS232 line are not used. However, the CP 5411 is not included in this documentation.
Supplementary literature (for emergency situations and additional applications!):	
7	User Documentation SIMATIC TDC
8	Manual on the COM PROFIBUS parameterizing software
9	Manuals of the other nodes: SIMATIC S5, ET 200U, ET 200B, SIMOVERT Master Drives

Table 3-25 Legend, hardware and software components for SIMATIC TDC

3.8.3.6 Configuring under STEP 7 CFC

General

In order to simplify unified configuring of a "PROFIBUS DP coupling" under CFC, the bus-specific CFC blocks are now grouped together and the relevant syntax explained.

When configuring an CP50M0 communications interface under CFC, the following should be observed:

- Precisely one central block @PRODP must be used for each CP50M0 communications interface
- A maximum of one transmitter- and/or one receiver block per communications partner

- Permitted communication utilities:
  - process data
  - parameter processing of variable-speed drives
- Permitted data transfer mode: Refresh (for receivers, also multiple)
- A maximum of one synchronization function block SYNPRO per CP50M0 communications interface
- A maximum of one diagnostics function block DIAPRO per CP50M0 communication module

Function blocks

Central block PROFIBUS DP coupling @PRODP

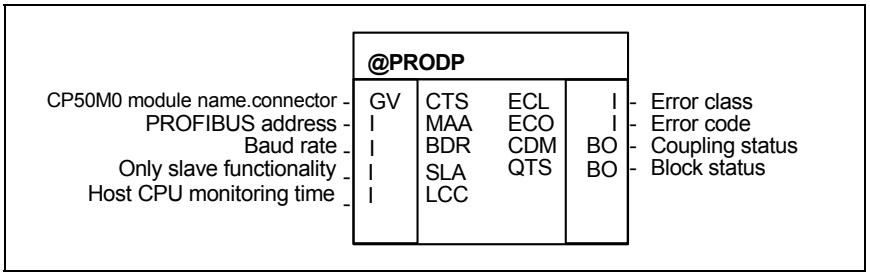


Fig. 3-33 Central block, PROFIBUS DP coupling @PRODP

- **Use**
  This function block initializes and monitors the PROFIBUS DP coupling (CP50M0). It may only be configured in a sampling time of 32 ms ≤ TA ≤ 255 ms.
- **I/O**
  ECL, ECO, CDM, QTS and YTS are service- and diagnostics I/O which are generally used for SIMATIC TDC start-up (commissioning). They are not used for configuring.

Further information
on the I/O of the central block PROFIBUS DP coupling @PRODP, refer to the User Documentation " SIMATIC TDC, function block library".

CTS	The configured name of the CP50M0 module (identical with the entry in the master program, actual: D04) and the designation of the interface (X01 or X02, actual: X02) is specified at this initialization input.
MAA	Just like all of the other bus nodes, the CP50M0 interface has a station address. This must be specified at this connector (a number between 1 and 123, actual: 4).
BDR	The baud rate, which the CP50M0 interface uses on the bus, is set using this connector. This value must be specified in a code: 0=9,6 kbaud ; 1=19,2 kbaud ; 2=93,75 kbaud ; 3=187,5 kbaud ; 4=500 kbaud ; 5=1,5 Mbaud ; 6=3 Mbaud ; 7=6 Mbaud ; 8=12 Mbaud ; (actual: 5).
SLA	Initialization input, only for slave functionality: 0: CP50M0 operates as PROFIBUS master and/or slave. A COM PROFIBUS database must be loaded. 1 or 2: SS52 operates as pure PROFIBUS slave without COM PROFIBUS database 1: Slave with either inputs or outputs, 2: Slave with inputs and outputs (actual: 0)
LCC	Initialization input for the time in which the CP50M0 monitors the SIMATIC TDC host CPU: <0: No monitoring 0...10: Monitoring time=1s (default) >10: Monitoring time in 1/10 s (actual: 0)

Table 3-26 I/O of the central PROFIBUS DP coupling block

3.8.3.7 Using transmit- and receive blocks

General

The function blocks of the communications utility, process data must be configured for PROFIBUS DP.

The address connections AT and AR of those blocks, which access the CP50M0 data interface, must have the following syntax (rules):

AT/AR- 'Channelname.Addressstage1.Adressstage2'

Channel name

- Must be unique, corresponding to the general communication rules (the channel names of all transmit- and receive blocks, which access the same CP50M0 communications interface, must be different)
- It may consist of a maximum of 8 characters
- It has no special significance for PROFIBUS DP

Address stage 1

- The PROFIBUS address of the communication partner is specified in this address stage.
- Using address 0, this channel goes to the slave and is called-up by other bus masters.
- External slaves can be addressed using addresses 3..123.

- A PROFIBUS address may only be used once for each transmit-/receive channel.

Address stage 2
This address stage is configured with one or two characters:

- **1st character:** Defines the byte order to transfer word quantities for various communication partners.
  - **1=Motorola format** (high byte before the low byte)
    Thus, it corresponds to the telegram structure of the PROFIBUS standard, and should be used as standard, especially when transferring single word quantities to standard bus nodes (analog I/O, SIMOVERT, SIMATIC etc.)
  - **0=Intel format** (low byte before the high byte)
    Can be used for data transfer to devices where data is processed according to the Intel format just like in SIMATIC TDC (e. g. second CP50M0)

Coupling partner	1st character
SIMOVERT Master Drives with CB 1 (standardized bus nodes)	1
ET200 distributed periphery (standardized bus nodes)	1
SIMATIC (IM 308 C,...) (standardized bus nodes)	1
SIMOREG 6RA24	1
MICRO / MIDI Master (standardized bus nodes)	1
SIMATIC TDC (CP50M0) (the coupling partner must also have the same setting)	0

Table 3-27 Byte order for various communication partners

- **2nd character** (optional, only for receivers):
  When an "R" is entered at a receive channel, the CP50M0 is authorized to read other slaves (shared input).

3.8.3.8 Configuring the typical configuration in CFC

General

In this case, it does not just involve process data processing, but mainly in implementing the listed communication paths to the other bus nodes.

A CFC chart with explanations shows how to configure the PROFIBUS DP. The CFC chart does not purport to include all details.

The following are to be configured:

- CPU CPU550 in slot S01 under the name D01P01:
- Communications module CP50M0 at slot S02, designation D02
- Communication interface 1 on CP50M0 connector X01

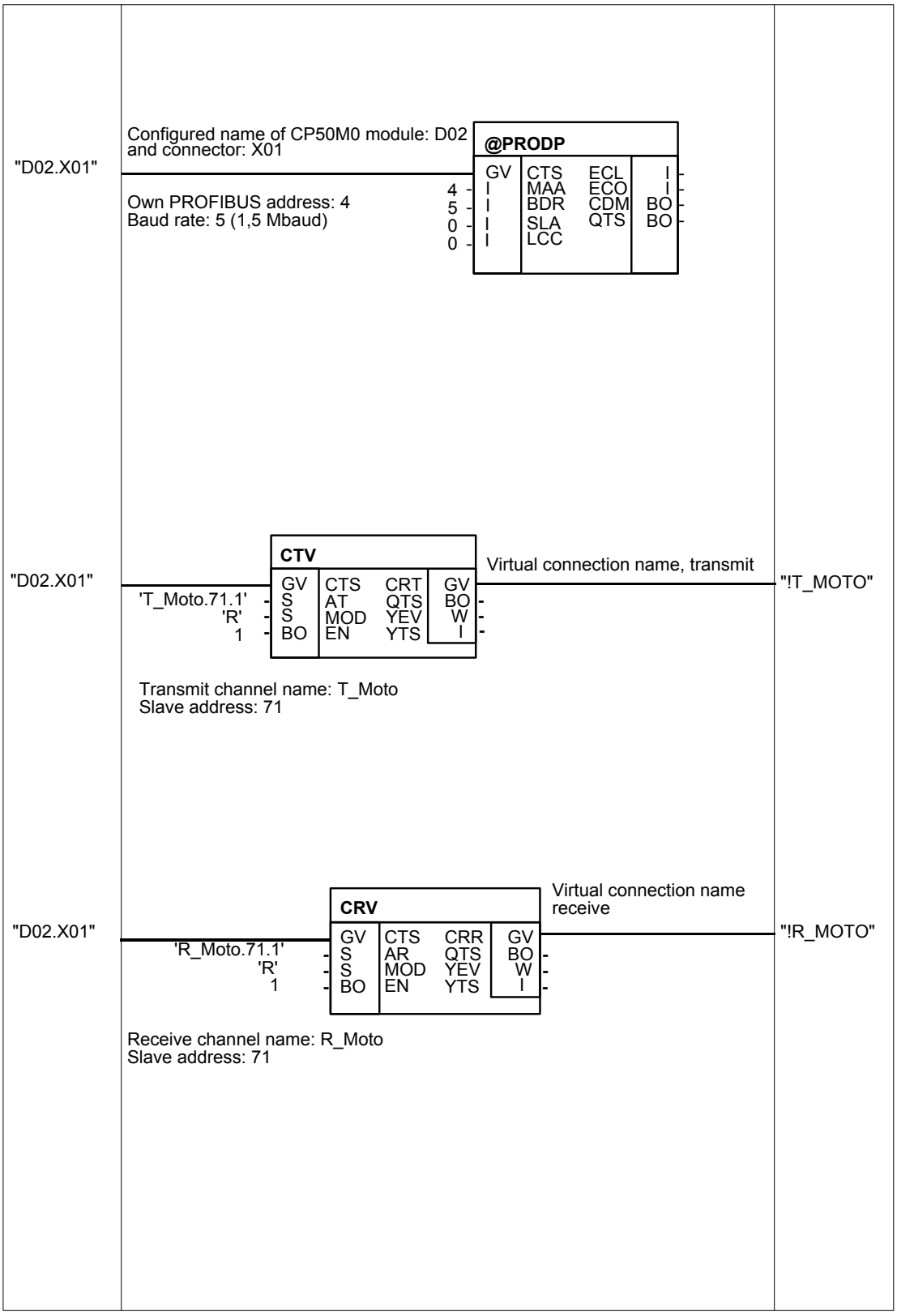


Fig. 3-34 CFC chart (Part 1) of the typical configuration

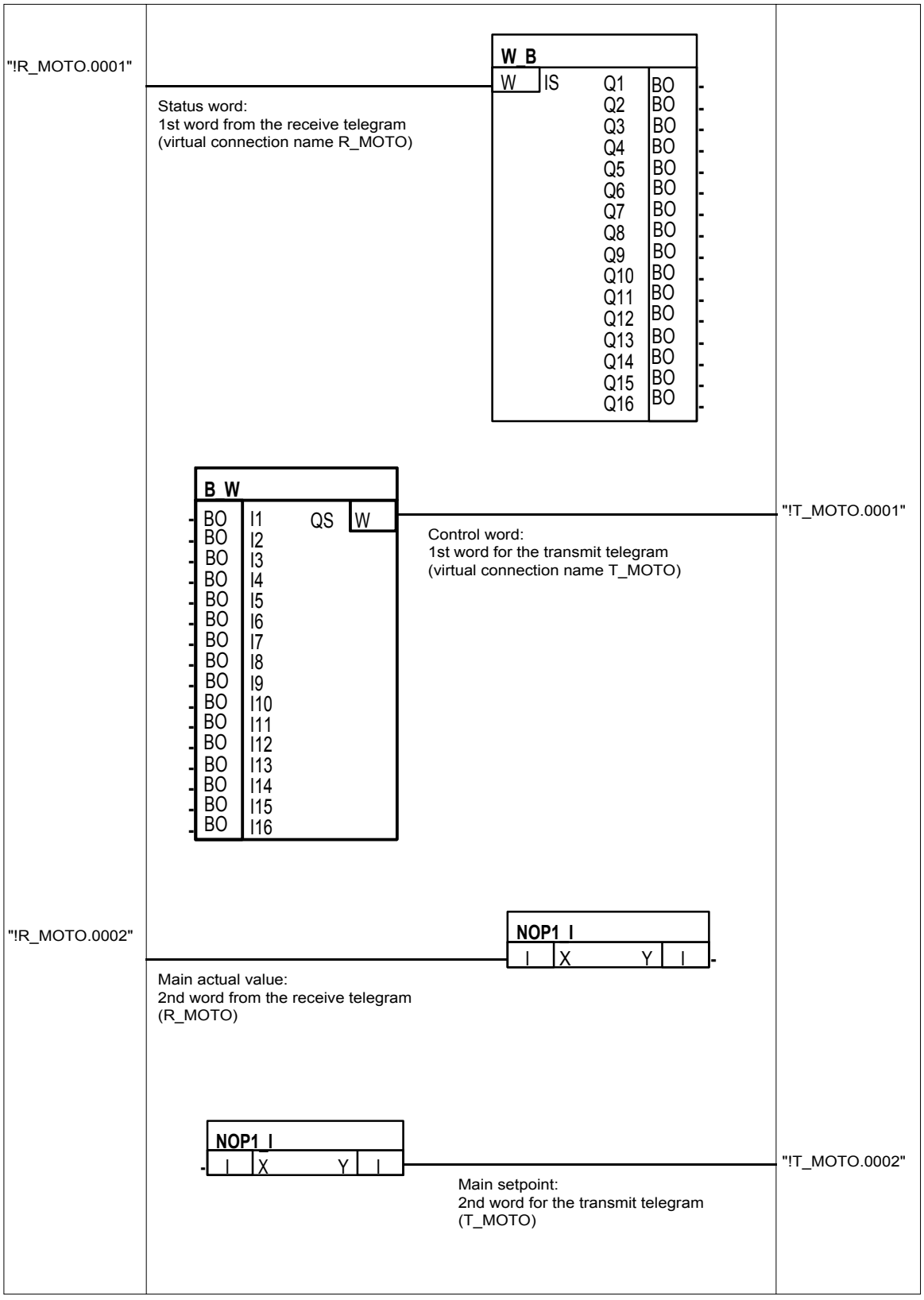


Fig. 3-35 CFC chart (Part 2) of the typical configuration

3.8.3.9 Configuring the SS52 communications module with COM PROFIBUS

General

If a communication interface was configured of the CP50M0 (currently: X01, then values are transferred between the transmit- or receive blocks and the bus connector on the CP50M0 communications module. As SIMATIC TDC is a freely-configurable system, the following logical communication structures must be assigned:

- Bus parameters defined (baud rate, ...)
- The communication associations between the nodes defined (who communicates with whom, and in which function?)
- The communication objects must be defined (communication objects are useful (net) data. For SIMATIC TDC they consist of the process- and device data. However, for the typical configuration, communications only involves the process data.)

This data (in the following, designated as COM database) is saved on the CP50M0 in a permanently integrated memory and are changed and adapted by downloading via the 9-pin sub-D connector of the module.

3.8.3.10 Generating the COM database with COM PROFIBUS

Procedure

Master and slaves of a bus structure are configured using a graphic user interface and a list of communication partners which are supported..

At the start, all communication associations of the typical configuration are defined by selecting the nodes involved.

Parameterizing the 1st host system

1. After the program start, the first master system is set-up using the menu command **File > New**.

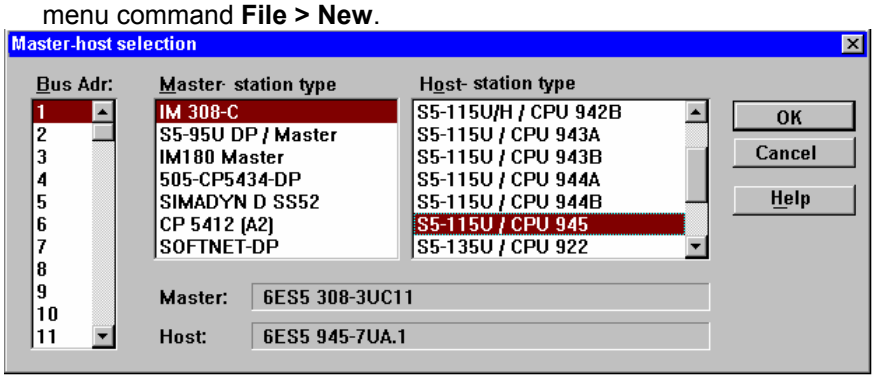


Fig. 3-36 Dialog box, "Master-host selection"

2. After buffering the data (**File > Save under...**) using any name (current: "Typical"), a first host system is generated with the name "Mastersystem <1>". The code number (current: 1) is identical with the selected PROFIBUS address. This first step defines who has the “say“ on this host system.

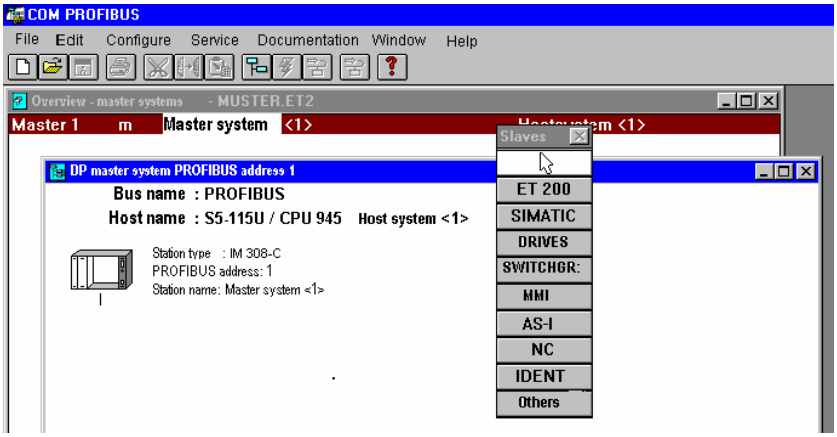
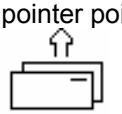


Fig. 3-37 Window "DP master system PROFIBUS address 1"

3. After selecting the button "ET200" in the "Slaves" menu, the mouse pointer points to an empty box with an arrow upwards.



This allows slaves to be assigned to the S5 station, by positioning the mouse pointer under the station symbol and then clicking on the mouse.

4. After interrogating the PROFIBUS address (current: 4) the communications partner can be selected in an additional selection window.

Fig. 3-38 "PROFIBUS address" window

5. The majority of the setting possibilities in the "Slave characteristics" window are of now significance for the typical configuration. The standard settings can be used. Only the family (current: SIMADYN), the station type (current: "SS52 master/slave") and the "Configuring..." button are important.

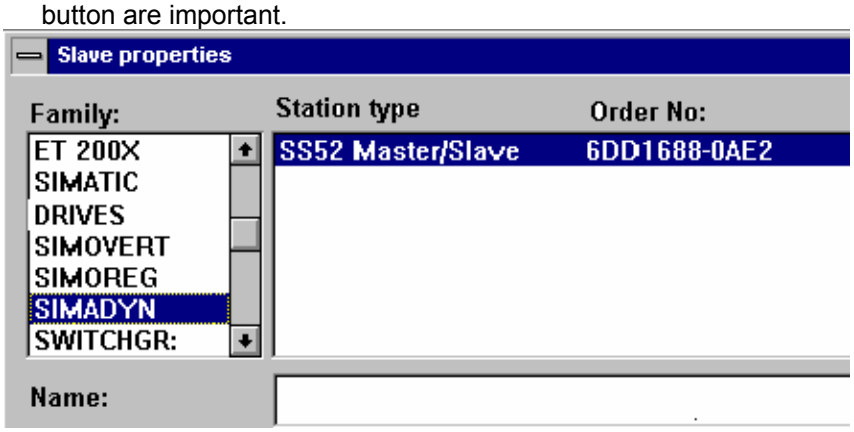


Fig. 3-39 Dialog box, "Slave characteristics"

6. However before configuring starts, the specified settings must be acknowledged with OK in a dialog box "Master-host selection".

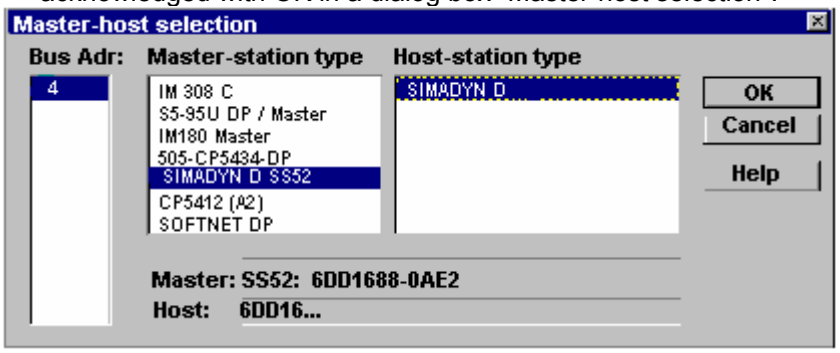


Fig. 3-40 Dialog box, "Master-host selection"

7. The bus node is now actually configured.
   For the CP50M0 communications module, this configuration window is at first completely empty. The net data structures must now be entered in the list in the dialog box "Configuring: SIMADYN D slave ...".

NOTE S5 is the master in this "Master system <1>" so that the transmit- and receive mode must be considered from its perspective (I/O addresses of the S5).

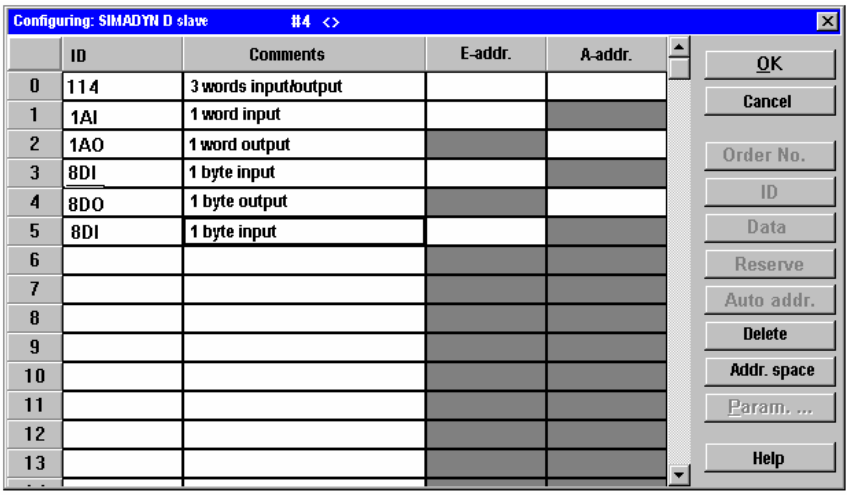


Fig. 3-41 Dialog box, "Configuring: SIMADYN D slave ..."

8. All of the data types are entered in the "ID" column. In this case, the associated dialog box must be activated. You can achieve this by either double clicking on a cell, or after highlighting the cell, depressing the "ID" button.
   The following parameters can now be specified:

- **Type**
  Select between:
  - input, output
  - input/output
  - empty location
  - special format
- **Length**
  1 to 16
- **Format**
  Select either single-word- or byte format

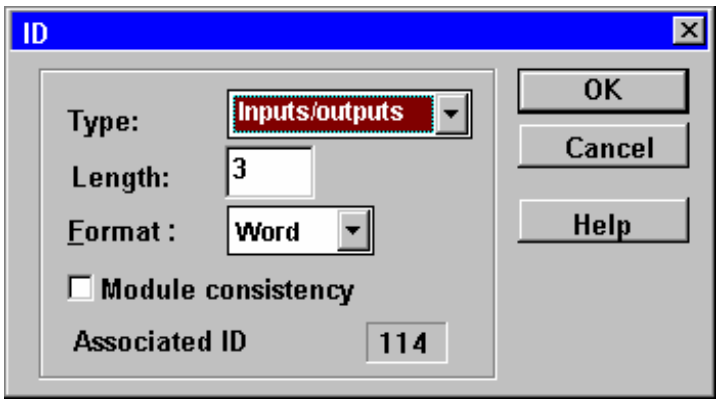


Fig. 3-42 Dialog box, "ID"

9. After terminating the dialog with OK, the appropriate ID is entered in the list. The sequence of the process data in the telegram is defined by the position at which the ID is entered in the input or output address ranges (fields with a grey background are not taken into account). Entries into the comments column are optional and can be freely configured. The address settings ("I-Adr." and "O-Adr.") are not required for the SIMATIC TDC/SIMADYN D database.
Thus, the first host system has been generated in which the SIMATIC TDC is slave to the S5. Parameterization has now been completed. It should be observed, that this involves the configuring data for the IM308 (S5); therefore these no longer have to be processed, as they are not relevant for the CP50M0.

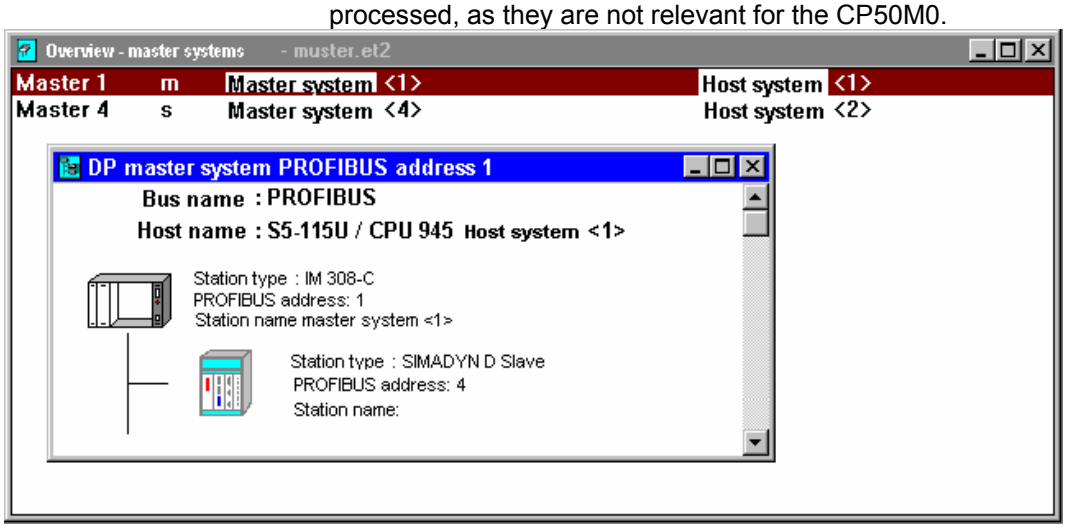


Fig. 3-43 Window, "DP master system PROFIBUS address 1"

Parameterization of the 2nd host systems

1. The first host system is closed by double clicking on "Master 4" and the second host system is made accessible so that the CP50M0 master can be parameterized.

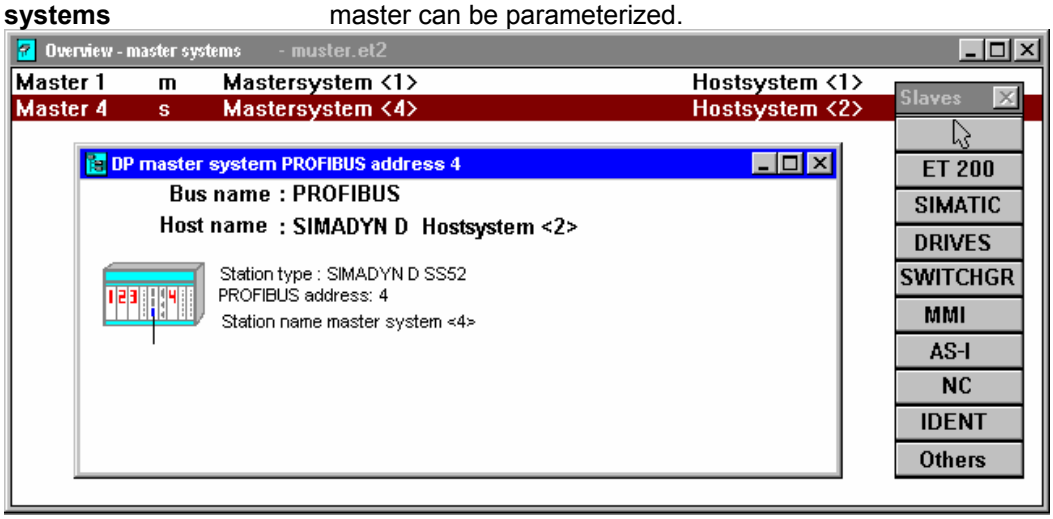


Fig. 3-44 Dialog box, "Overview, master systems"

2. By double clicking on the symbol "SIMADYN D" it can be seen how important it is to first set-up the SS52 as slave in the "Host system <1>". The complete telegram structure is automatically transferred into

the CP50M0 configuration with the difference that the telegrams lie interchanged in the address ranges: The output of S5 becomes the input for SIMATIC TDC and vice-versa.

The data identification (by configuring...) now has a grey background and, for this communication, can no longer be changed from the present host system (identification and comment belong to the S5). The data are acknowledged with "OK". Thus, communications to the S5 have been set-up.

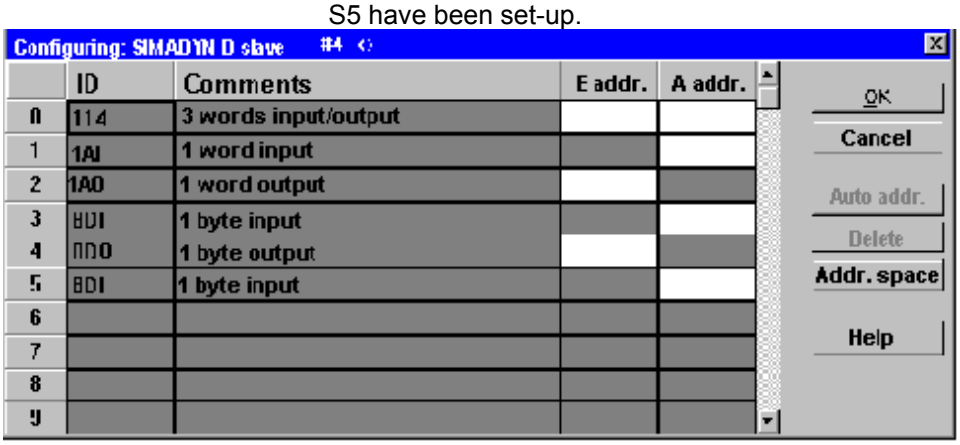


Fig. 3-45 Dialog box, "Configuring: SIMADYN D slave"

3. The master functions of CP50M0 can be configured. To realize this, you must return to the DP master system window, PROFIBUS address 4. After the slave menu has been re-activated (the mouse pointer changes), ET 200 U, ET 200 B and SIMOVERT Master Drive are coupled one after the other. Each time a component is called-up, you are prompted for the PROFIBUS address. The "Slave characteristics" window then automatically opens, in which, as already described, the necessary settings can be made using **Configuring...**.
4. As the field devices are pure slaves, depending on the function, type of construction and "Intrinsic intelligence" they can only be parameterized with some restrictions.
   The individual configurations are as follows:
   - **ET 200 U**
     Modular structure with three output modules (each with 8 digital outputs) and one input module (8 digital inputs): Three bytes must be transmitted and one byte must be received.

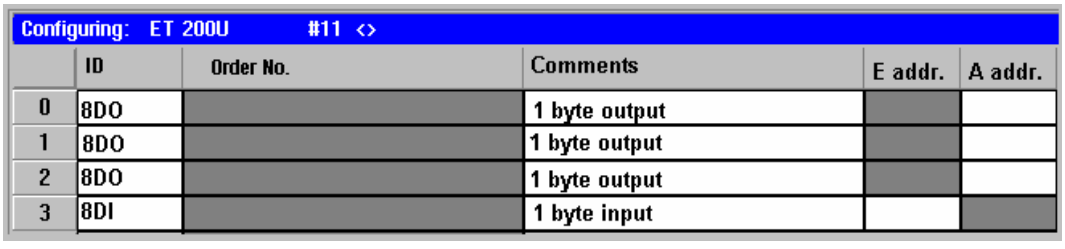


Fig. 3-46 "Configuring" window

– **ET 200 B**
Compact type of construction with eight digital outputs and eight digital inputs: One byte in the transmit telegram and one byte in the receive telegram. The IDs are specified by the module selection.

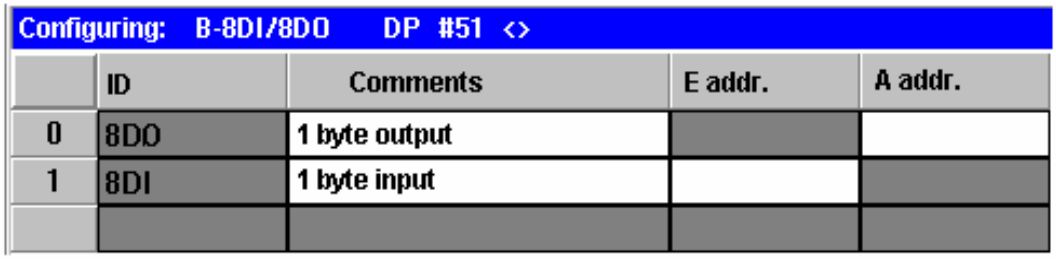


Fig. 3-47 "Configuring" window

– **SIMOVERT Master Drive**
Slave with intrinsic intelligence: Depending on the drive converter setting, five different telegram structures (PPO types) are permitted. These must be defined when configuring and can no longer be changed. (Fields have a grey background and are therefore inactive)

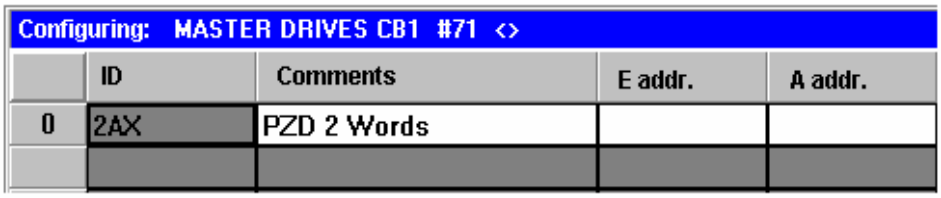


Fig. 3-48 "Configuring" window

5. After configuring has been completed, the display should look like this:

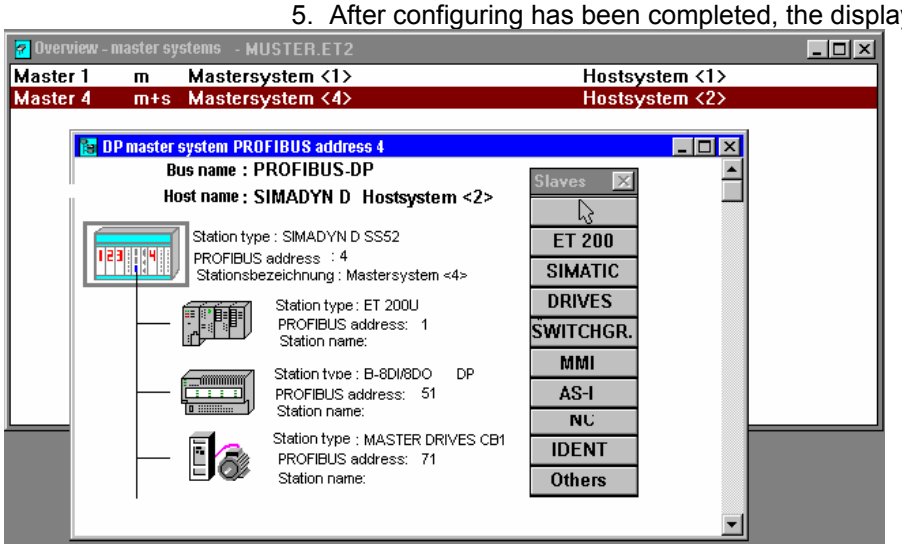


Fig. 3-49 Window, "DP master system, PROFIBUS address"

Changing the slave configuration

The configuration of the individual slaves can be subsequently changed and adapted.

1. Select the particular symbol in the display above by clicking on it twice with the mouse. You can return to the configuration dialog boxes via the "Slave characteristics" window.
2. To complete parameterization, the bus parameters must be set as a final step. A dialog window is opened under **Configuring > Bus parameters...** In this window only the bus profile (PROFIBUS-DP) and the baud rate are of importance for this typical configuration. The baud rate must coincide with that specified in the CFC, and in our example is limited by the ET 200 U and the CB1 communications module of the SIMOVERT Master Drive (currently: 1.5 Mbaud).

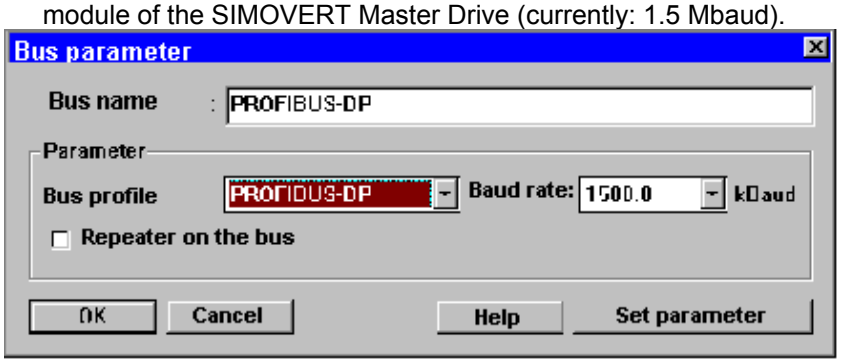


Fig. 3-50 Dialog box, "Bus parameters"

3. This completes the configuration of the CP50M0 for this typical configuration, and it can now be saved.

Transferring the configured software into the CP50M0 memory

The next step when setting-up the CP50M0 configuration is to transfer this configured software into the memory of the CP50M0 communication interface. There are two ways to do this:

- Transfer data via a second module interface (RS232) which is located on the same 9-pin sub-D connector as for the RS485.
  - Data transfer via RS232 can be executed using a standard PC interface (COM 1 or COM 2) whereby a special transfer program named "SS52load" downloads data into the CP50M0 memory.
  - This download requires the "2bf" file format. This is why the marked "Host system <2>" must be converted into the correct format via the menu command **File > Export > Binary file...** (the host systems must be separately handled for this operation). The CP50M0 configuration file is thus now located in the root directory of the COM PROFIBUS program in the directory "\progdat" for transfer to the module.
- Transfer data via the PROFIBUS interface RS485 (is directly supported by COM PROFIBUS).
  - Transfer via RS485 is not discussed, as a special PC interface card (e. g. CP 5411) is required in this case.

3.8.3.11 Downloading the COM database into the CP50M0

Hardware required

The following hardware is required to download the COM database onto the CP50M0:

- RS232 connection between the PC and SS52
  - In addition to the RS485, an additional RS232 interface is also integrated on the 9-pin sub-D connector.
    Further information
    on the sub-D connector, refer to the User Documentation "SIMATIC TDC, hardware description".
  - A special cable (TxD to RxD) must be assembled as the pin assignment of this connector does not correspond to any specific standard.

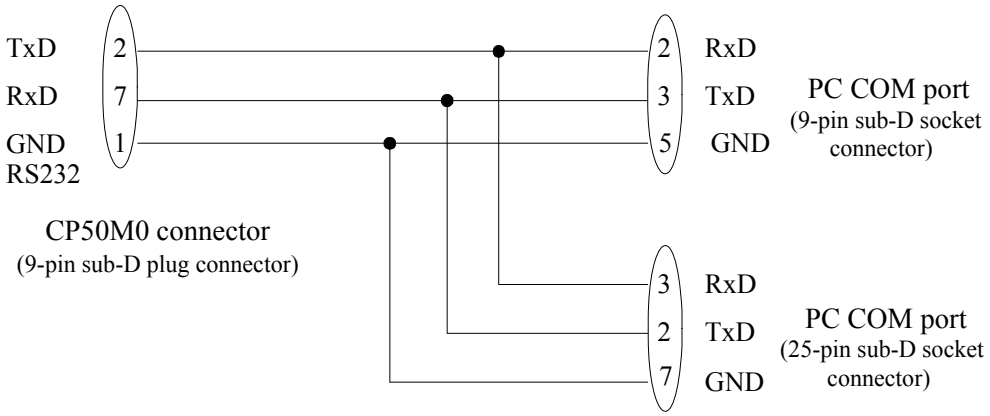


Fig. 3-51 RS232 interface

3.8.3.12 Working with the "SS52load" download tool

SS52load

SS52load is integrated in COM PROFIBUS (from Version 3.1).

The user interface offers the following functions:

- **Option comport:** Defines the COM port to be used
- **File download:** Selects the required file and downloads it

3.8.3.13 Behavior of the CP50M0 during and after the download

General

In order to successfully download, the different behavior patterns of SIMATIC TDC and the CP50M0 communication modules should be known before, during and after this operation. General system conditions are output via a green and a yellow LED, which are provided at each of the two CP50M0 interfaces.

These LEDs only provide information as to whether the SIMATIC TDC as self-contained autonomous system is functioning correctly, or if there are faults/errors. Bus activities or communications with other bus nodes are not evaluated.

LED statuses when SIMATIC TDC runs-up

- When the power is applied, both LEDs briefly light-up (approximately half a second).
- The yellow LED then goes dark, so that only the green LED is lit during the remaining run-up time (approx. five seconds). Downloading is not possible during this time.
- After the run-up phase has been completed, the operating status of the CP50M0 is displayed.

3.9 MPI coupling

3.9.1 Characteristics and hardware

Characteristics

MPI (Multi Point Interface) is the standard communications protocol for SIMATIC S7/M7. Data transfer is realized via a multi-master bus with a maximum of 126 nodes.

For SIMATIC TDC, MPI is used to connect the CFC for start-up and testing configured software, and is also used to communicate with WinCC and SIMATIC OPs.

With the MPI coupling, the communication utilities service (FB-SER) and S7 communications (FB-S7OS) are used.

Hardware

The following hardware is required for the MPI coupling:

- Subrack
- CPU
- CP50M1-/CP5M0 module (corresponding interface in HWKonfig for MPI configuring, at CP50M1 only X01)
- MPI cable (is in the scope of delivery of the PG contain)

3.9.2 Configuring

HWConfig

The CP50M1/CP50M0 communications module and the SS52/MPI communications module must be configured in HWConfig. Its own MPI address must be specified for ES.

Function block @MPI

Precisely one @MPI central coupling block must be configured for each SS52/MPI. The @MPI function block initializes and monitors the MPI coupling.

Additional information
to configure an MPI coupling, refer to:

- Section "Communications Utility Service"
- Section "Communications with SIMATIC Operator Panels"
- Section "Communications with WinCC (MPI)"

3.10 Table function

3.10.1 Introduction

The table function in SIMATIC TDC / SIMADYN D provides the user with the possibility of linking-in and using tabular values (values in a table) in a configured software application. In this case, the function blocks TAB and TAB_D must be configured on the SIMATIC TDC and SIMADYN D sides. Tabular values, data type REAL are managed using the TAB and data type DINT, using TAB_D. The user provides the tabular values.

The table function can be configured in three modes:

- **Manual mode**, i.e. the tabular values are directly entered at the block via an online interface (e.g. CFC in the test mode), or transferred to the block using teach-in from the program.
- **Automatic mode: Communications**, i.e. the tabular values are transferred via a communications interface (TCP/IP, DUST1, S7 via P bus). In order to transfer tabular values from an S7 control to a SIMATIC FM 458 application module via the P bus, in addition, the WR_TAB should be configured on the S7 control side.
- **Automatic mode: Memory card**, i.e. the table values are downloaded into the memory card, from where they are read.

NOTE The "Automatic mode, memory card" mode is presently still not available.

It should be noted, that it is only possible to toggle the modes between "**Manual mode"** and **"Automatic mode: Communications"** as well as "**Manual mode"** and **"Automatic mode: Memory card"**.

A validity check is made if the tabular values have been entered or transferred. The address of the table is displayed at "TAB" output.

The tabular values are managed twice, i.e. in two tables. The table, defined as "valid" (=active) is used for all arithmetic/computation operations of the configured application software. The "invalid" (=inactive) table is used to manage value changes. All of the tabular values, changed by the user, are initially transferred into the invalid table. If the inactive table is activated, the new tabular values are mirrored in the second table. The table which had been active up until then automatically becomes invalid. This means that the new tabular values are available in both tables.

Both tables can be saved in the SAVE area which is backed-up (buffered) by a battery in order to prevent data loss (connection SAV=1 when initializing).

NOTE A precise description of function blocks TAB and TAB_D is provided in their respective online help.
A detailed description of the WR_TAB function blocks is provided, further below in the Section "Function block WR_TAB".

3.10.1.1 Overview, "Manual mode"

The principle procedure in the "Manual mode" is shown in the following diagram:

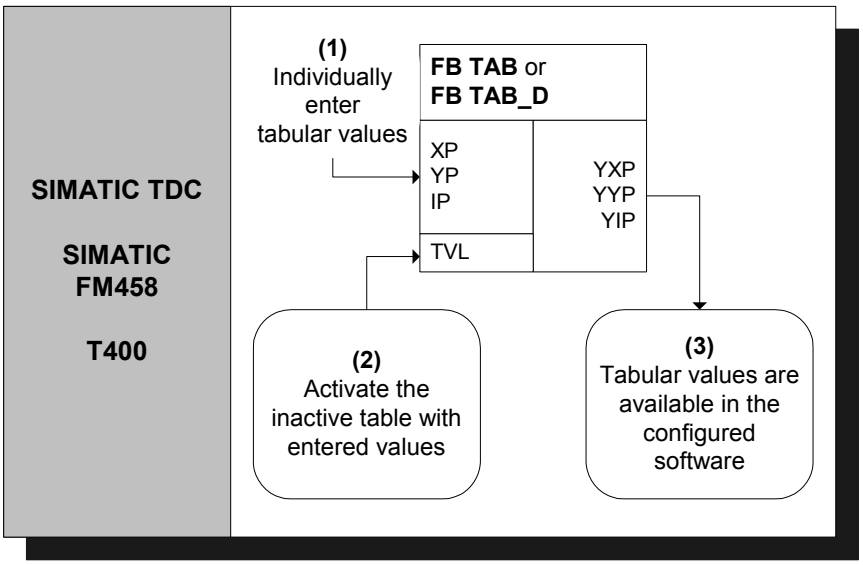


Fig. 3-52: Principle procedure in the "Manual mode"

A detailed description of the "Manual mode" is provided in Section "Manual mode" (Page 3-107)

3.10.1.2 Overview, "Automatic mode: Communications"

In the "Automatic mode: Communications", tabular values can be transferred using the following communication versions:

- S7 via the P bus for SIMATIC FM 458 (it is necessary to additionally configure the WR_TAB on the control side)
- TCP/IP (tabular values can be transferred from a SIMATIC TDC module to another one using the CTV and CRV FBs)
- DUST1 (tabular values can only be transferred via a DUST1 interface)

The tabular values are transferred using data telegrams.

The following diagram illustrates the principle procedure in the "Automatic mode: Communications" for transferring tables from an S7 control to a SIMATIC FM 458 application module via the P bus:

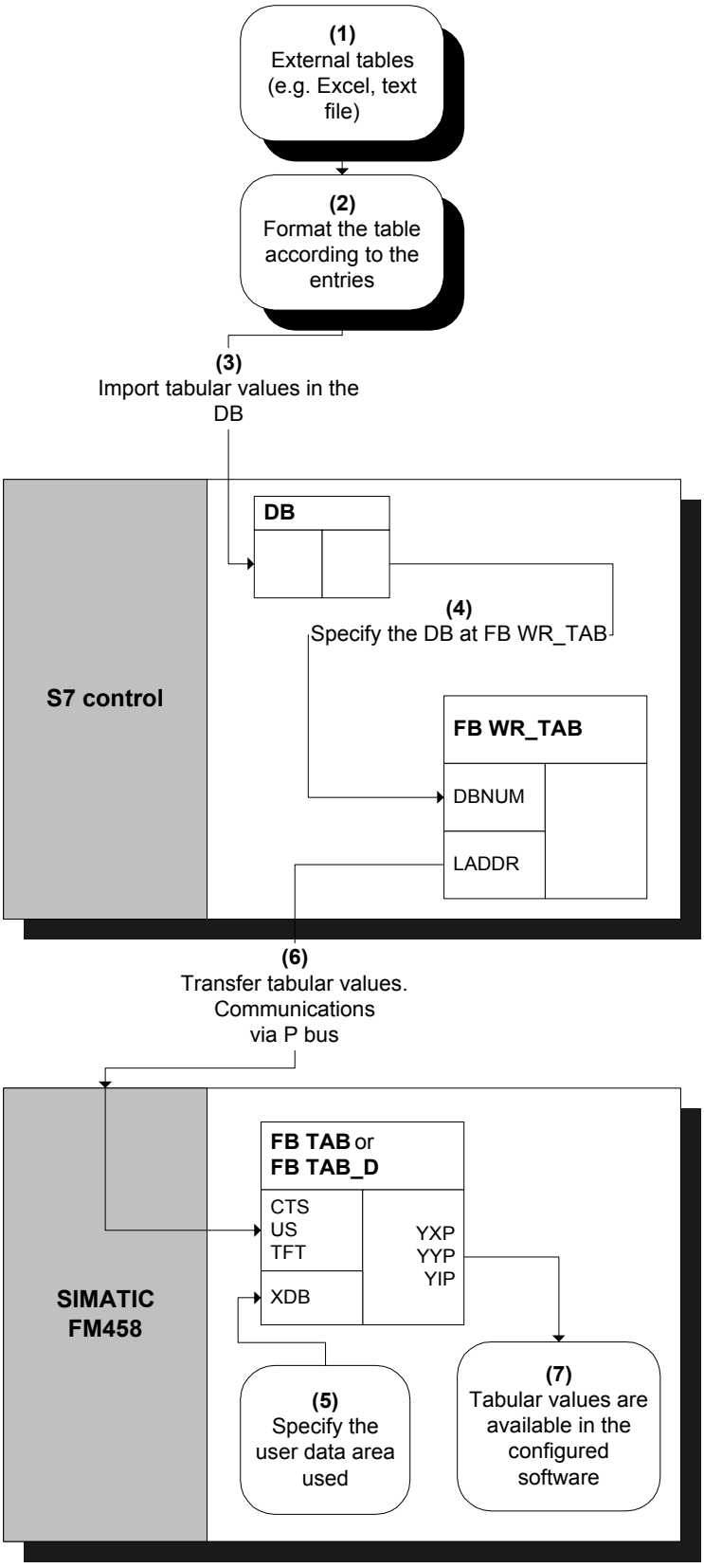


Fig. 3-53 Principle procedure for "Automatic mode: Communications"(via P bus)

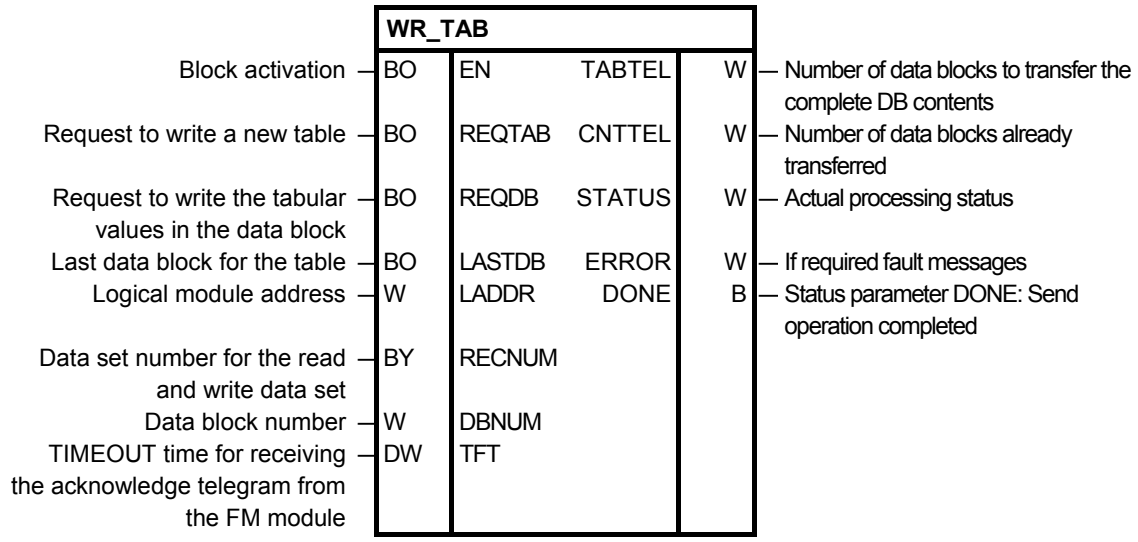
A detailed description of the "Automatic mode: Communications" mode to transfer tables from an S7 control to a SIMATIC FM 458 application module is provided in the Section "Automatic mode: Communications" (Page 3-109).

3.10.1.3 Function block WR_TAB

The function block WR_TAB is used to transfer tables from one S7 control to a SIMATIC FM 458 application module. The tabular values (permissible data types are REAL and double integer) are saved in a data block. They are transferred from WR_TAB to the function blocks TAB and TAB_D on the SIMATIC FM 458 application module, which then internally manages the tabular values.

The WR_TAB should be configured on the control side. The tabular values are transferred from one S7-400 control to a SIMATIC FM 458 application module via the P bus. All of the values are always transferred, which are in the DB specified at the DBNUM input.

Symbol



I/O

The individual connections (I/O), their data types and a connection description are listed in the following table:

Parameter	Declaration	Data type	Description
REQTAB	INPUT	BOOL	REQTAB = 1: Request to write a new table
REQDB	INPUT	BOOL	REQDB = 1: Request to write the tabular values which are saved 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 set number for the read and write data set
DBNUM	INPUT	WORD	Data block number of the DB in which the tabular values are located.
TFT	INPUT	DWORD	TIMEOUT time in ms for receiving acknowledge telegrams from the SIMATIC FM 458 application module.
TABTEL	OUTPUT	WORD	Number of data blocks required to transfer the complete DB contents
CNTTEL	OUTPUT	WORD	Number of data blocks already transferred to the FM module
STATUS	OUTPUT	WORD	Indicates the current status of the processing / data transfer: 0: Table transfer is inactive 1: Table transfer is active. Table values have been partially transferred from a DB (wait for the next partial transfer) 2: Table values have still not been completely transferred from a data block.
ERROR	OUTPUT	WORD	If a fault/error occurs while processing the function, then the return value is an error code
DONE	OUTPUT	BOOL	Status parameter DONE=1: Send operation has been completed

The following errors can occur and are displayed at the ERROR output:

Error code	Explanation	Remedy
0xB210	OK	-
0xB211	Logical module address invalid	Specify a valid module address at input LADDR.
0xB212	Data set number not valid	Enter the tabular values in an increasing sequence in the DB.
0xB213	Invalid table data format	Tabular values must have data type REAL for the TAB and data type DINT for the TAB_D.
0xB214	The data format of the new data set does not match that of the previously transferred data set	Ensure that all of the tabular values have the same data format.
0xB215	FM 458 does not respond	Check the communications connection and configuring.
0xB216	Table is too large	Transfer the table in sub-sets, i.e. either distribute tabular values over several DBs or after each partial transfer write new (additional) tabular values into DB and transfer.
0xB217	Table is not complete (X / Y values)	Complete the table, there must be a Y value for each X value.
0xB218	REQTAB is reset during processing	Transfer the tabular values again.
0xB219	REQDB reset during processing	Transfer the tabular values again.
0xB21A	DB number is not valid	Specify a valid DB number.
0xB21B	TIMEOUT when receiving the acknowledge telegram	Check the communications coupling and configuring. Transfer the tabular values again
0xB21C	Invalid processing status	Check the configuring of the WR_TAB.

Errors associated with the SFC58 or SFC59 are displayed at the ERROR output.

3.10.2 Manual mode

3.10.2.1 Application

The "Manual mode" mode represents the simplest way of inserting tabular values into a configured software package. However, it is comparatively time consuming as data has to be manually entered or taught-in from the program.

Entering tabular values

After the TAB or TAB_D has been correctly configured, the tabular values can be entered one after another. To start off with, the table size, i.e. the number of value pairs (=points) should be specified at input NP. If the table is to be saved in the SAVE area, then input SAV of the must be 1.

The tabular values can then be subsequently entered. In this case, to start, the index point i should be specified at input IP of the value pair to be entered. The X and Y value of the point should then be entered at inputs XP and YP. In order to accept the entered value, after entering each value pair, input WR should be set from 0 to 1. Before entering the next point, the index at input IP should be incremented. The values for this point should then be entered. This procedure is repeated until all of the values have been entered.

A specific sequence does not have to be observed when entering the individual points.

The number of entered points must match the data at input NP.

All of the entries during this procedure are transferred into the inactive table of the and are only available after being activated in the configured software. In order to activate the inactive table with the entered values, input TVL should be set to 1.

Additional changes can then be again made in the inactive table and are only available after this has been re-activated again.

Interrogating the tabular values

In order to output the entered tabular values, after entering the data at input IP, the index of the point i, to be displayed is specified, and input RD is set from 0 to 1. The tabular values of point i are then displayed at the outputs YXP (X value) and YYP (Y value). The index of point i is output at output YIP.

3.10.2.2 Configuring

For the "Manual mode", only the TAB and/or TAB_D have to be configured depending on whether tabular values, data type REAL and/or DINT have to be managed. Each table may only contain values associated with one data type. If several tables having different data types are to be managed, then an TAB or TAB_D must be configured for each table.

The function blocks TAB and TAB_D should be configured in the same sampling time of 32ms. The following connection (I/O) settings are required:

AUT =	0 (automatic mode de-activated)
NP =	[specifies the table size]
XP =	[enters the X values]
YP =	[enters the Y values]
IP =	[enters the value pair to be changed]
TVL =	1 (to activate the table after all of the values have been entered)
WR =	1 (to transfer the value pair which was entered in the table)
RD =	1 (to display the value pair, specified under IP, at outputs YXP and YYP)

NOTE If, in the "Manual mode" the CTS connection is set to "0" when initializing (CTS=0; AUT=0), then it is no longer possible to changeover into the "Automatic mode: Memory card" (CTS=0; AUT=1).
If the CTS connection is set to "0" while initializing, and the "Automatic mode: Memory card" is activated (AUT=1), then it is possible to subsequently changeover to "Manual mode" (CTS=0; AUT=0). The table, saved on the memory card, can then be processed in the "Manual mode".
If, after this, a change is made back to "Automatic mode: Memory card" (CTS=0; AUT=1), this no longer has any effect, because it is only activated during the initialization operation.
If a communications interface is configured at the CTS connection, it is possible to toggle, as required between "Manual mode" and "Automatic mode: Communications".

3.10.3 Automatic mode: Communications

3.10.3.1 Application with an S7 control and SIMATIC FM 458 application module

Transferring tabular values

The following prerequisites must be fulfilled in order to successfully transfer tables:

- The function blocks TAB and/or TAB_D must be configured in the FM 458 application module corresponding to the configuring specifications for "Automatic mode: Communications" (A detailed explanation is provided in Section "Configuring for S7 control and SIMATIC FM 458 application module").
- The X and Y values of a table in a DB must always be present alternating. There must be a Y value for each X value, so that the number of values in a data set is always an integer number.

In order to start data transfer, inputs REQTAB and REQDB at WR_TAB must be set to 1. The tabular values of the DB, specified at input DBNUM at WR_TAB can then be transferred.

The actual number of transferred data blocks is always displayed at the CNTTEL output of the WR_TAB.

The number of data blocks is displayed at the TABTEL output of the WR_TAB, which is required until the complete contents of the DB are transferred to the SIMATIC FM 458 application module.

If the tabular values have been completely entered in the specified DB, or if it involves the last partial transfer of a table (sub-set of a table), which does not “fit” completely into a DB, then before starting the transfer, input LASTDB of the WR_TAB should be set to 1. This means that the SIMATIC FM 458 application module is signaled at the end of the data transfer. The STATUS output of the WR_TAB then changes from 2 to 0.

NOTE All of the tabular values, which are located in the DB, specified at the DBNUM input of the WR_TAB, are always transferred.

Table too large for a DB

If the table is too large for a data block, then the tabular values are split-up into individual sub-sets for transfer. The procedure is as follows:

To start, the first table section is written into the DB and is then transferred as described above. The LASTDB input of the WR_TAB remains at 0. The STATUS output of WR_TAB stays at 2 during data transfer and then changes, at the end of the table sub-set transfer (partial transfer) from 2 to 1.

The old tabular values in the DB should then be overwritten with the following tabular values. Once this has been completed, at WR_TAB the REQDB input should be again set from 0 to 1 to activate the next table sub-set transfer.

This procedure should be repeated until all of the tabular values have been transferred.

At the last sub-set transfer, input LASTDB of the WR_TAB should be set from 0 to 1. This signals the SIMATIC FM application module that data transfer has been completed. The STATUS output of the WR_TAB then changes from 2 to 0.

NOTE

If there is adequate user memory available, the table can also be saved in several different DBs. In this particular case, for each table sub-set transfer, only the matching DB number at the input DBNUM of the WR_TAB has to be specified. However, it should be ensured that the DBs are transferred in the correct sequence, so that all of the tabular values are transferred in an increasing sequence.

Data transfer duration

The time taken to transfer the tabular values depends on the following factors:

- Number of tabular values
- Size of the data blocks
- Sampling time of the TAB and TAB_D
- WR_TAB processing time

In each cycle, a telegram with 56 tabular values is transferred, from the control to the SIMATIC FM 458 application module.

The time taken for a table to be transferred can be calculated as follows:

Duration of the data transfer =	[No. of tabular values / 56] * cycle time of the slowest FB (i.e. TAB, TAB_D or WR_TAB)

The time taken for the data to be transferred via the P bus is not relevant for this estimation, as this data transfer time is generally less than 1ms and generally, the function blocks TAB and TAB_D are configured in sampling times which are greater than 32ms.

If a table is distributed over several data blocks, the time required increases. The reason for this is that in addition to the time taken to

transfer the tabular values, which can be determined using the formula above, the user has to manually make the changes described above.

3.10.3.2 Configuring for S7 control and SIMATIC FM 458 application module

The following function blocks must be configured for the coupling between an S7 control and an SIMATIC FM 458 application module via P bus:

- SIMATIC FM 458 application module:
  - TAB (for REAL data type) and/or
  - TAB_D (data type DINT)
  - @CPB (P-bus coupling, central block)
- S7 control:
  - WR_TAB

Each table may only contain values associated with one particular data type. If several tables with different data types are to be managed, then an TAB or TAB_D must be configured for each table.

WR_TAB is used to transfer the tabular values from SIMATIC DB to function blocks TAB and TAB_D. The tabular values are transferred using a data telegram. When the last data telegram has been transferred, the TAB or TAB_D is automatically signaled that all of the tabular values have been transferred and that the table should be activated. WR_TAB receives a checkback signal as to whether activation was successful or not. After the table was successfully activated, its address is output at the TAB output of the TAB or TAB_D.

TAB and TAB_D

TAB and TAB_D should be configured as follows:

They should be configured in a sampling time greater than or equal to 32ms. The following connection settings are required:

CTS =	[name of the configured communications interface]
AUT =	1 (automatic mode activated)
US =	[channel name.address stage1] (address data for receive)
MOD =	[data transfer mode] (H=Handshake; R=Refresh; S=Select; M=Multiple)
TFT =	[monitoring time in milliseconds] (maximum telegram failure time while receiving tabular values)
NP =	[specifies the maximum table size]

NOTE

If a communications interface is configured at the CTS connection, it is possible to toggle, as required between "Automatic mode: Communications" and "Manual mode".

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 set number for the read and write channels. This must be identical with "Address stage1" at the US connection of the TAB or TAB_D.]
DBNUM =	[specifies the data block number]

3.10.3.3 Inserting tabular values in the data block

In order to be able to transfer tabular values to a SIMATIC FM 458 application module, they must be available in a data block (DB). The DB should be programmed on the control side.

There are two ways of generating a DB with the required tabular values:

- Generating a new DB in STEP7 and manually entering the tabular values in the application "LAD/STL/CSF"
- Importing tabular values from an existing table (e.g. MS Excel) as external source in STEP7

3.10.3.3.1 Manually entering tabular values

In this case, it involves the simplest method of providing tabular values in a DB. It is realized by entering the initial (starting) and actual values of the individual table values manually in a newly generated DB in the application "LAD/STL/CSF". The steps required will now be explained.

NOTE

The initial value is any value which can be defined for every tabular value. It is only used if there is no actual value specified for the associated tabular value.
The actual value is that value which is made available as tabular value in the configured software. The required tabular values should be specified here.

(1) Generating a new DB under STEP7

To start, a new DB should be generated under STEP7. In this case, the "Blocks" folder is selected in the appropriate S7 program and in the context-sensitive menu, the entry "Insert new object → data block" is selected.

The procedure is shown in the following diagram:

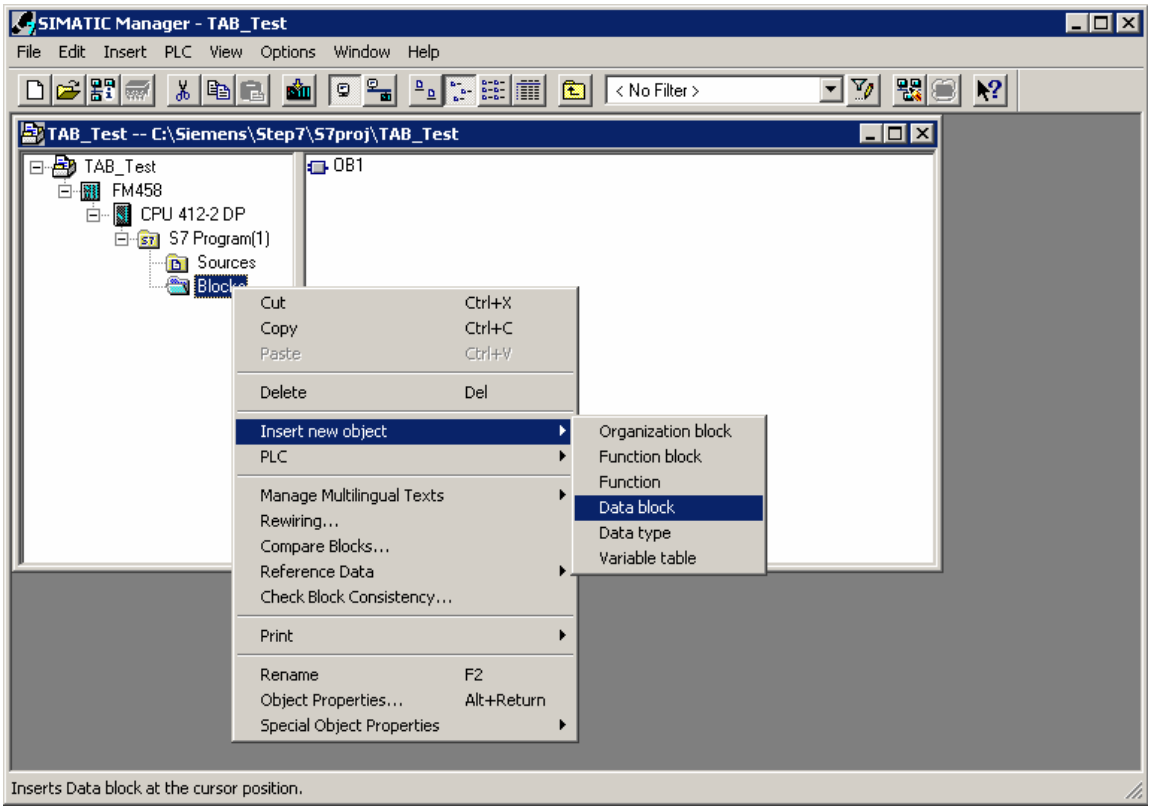


Fig. 3-54 Generating a new data block under STEP7

(2) Opening the new DB

The next step is to open the newly generated DB by double-clicking with the application "LAD/STL/CSF". "DB Editor" is the tool which is used to generate it and only one "Data block" is generated.

The following diagram illustrates the selection when opening a new DB:

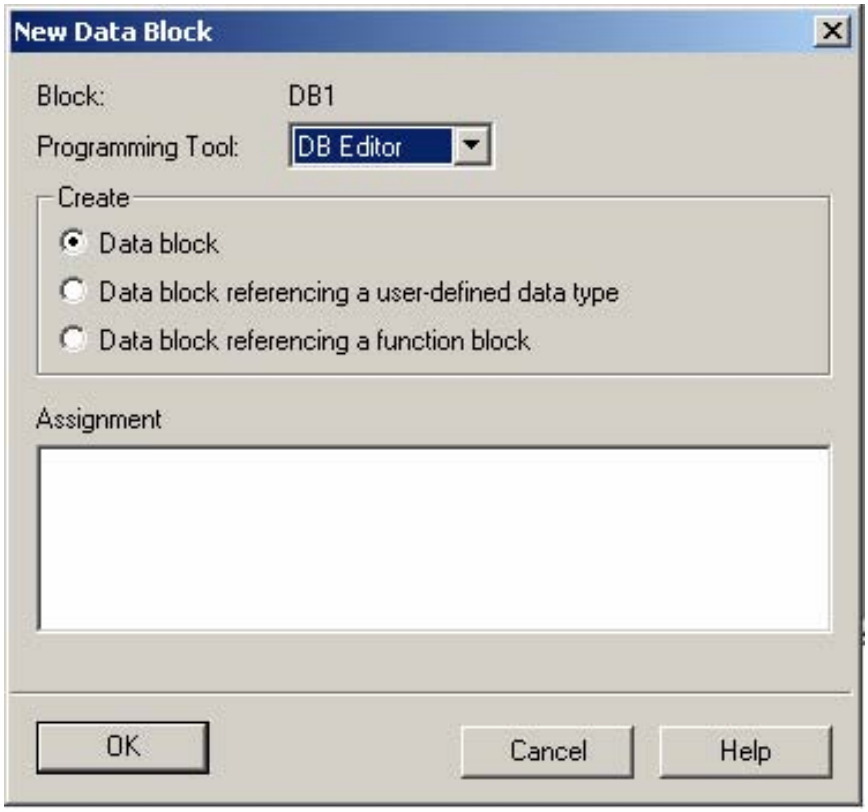


Fig. 3-55 Making a selection when generating a new DB

The opened, new DB is illustrated in the following diagram:

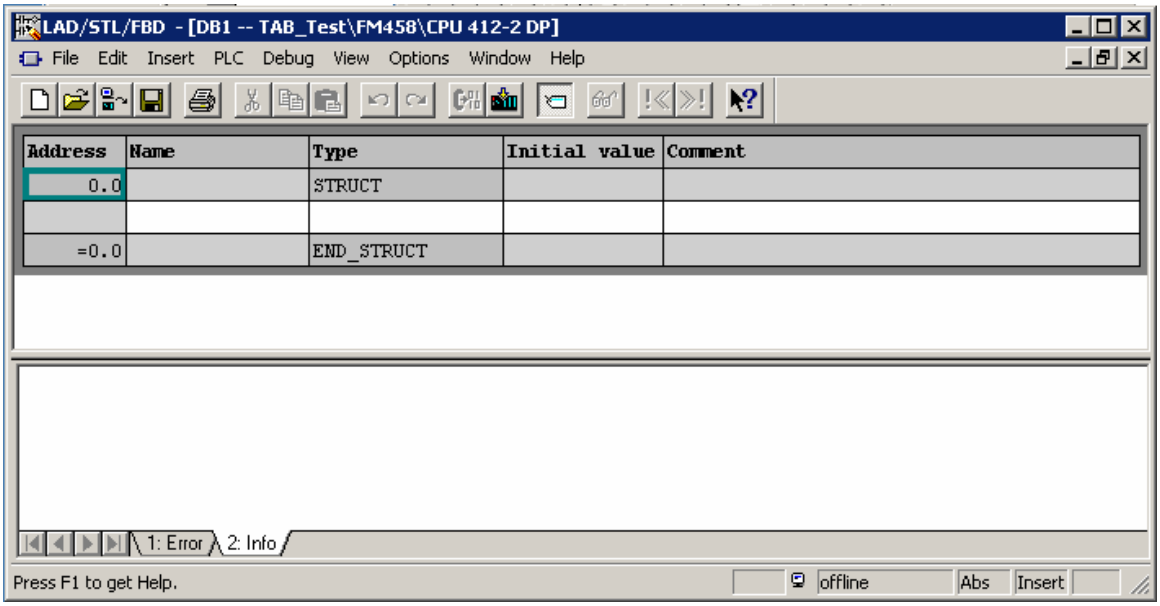


Fig. 3-56 Newly generated DB in the application "LAD/STL/CSF"

(3) Entering the tabular values

The required tabular values can now be entered. It should be ensured that the X and Y values are entered, alternating.

To start, the data type, used in the table, should be entered (REAL or DINT). In this case, the name is always "Data type", "WORD" type and initial value for data type REAL "W#16#1", for data type DINT "W#16#2".

Then, for each individual tabular value, the name, data type ("Type" column) and value ("Initial value" column) should be entered.

The procedure when entering tabular values, data type REAL, is shown in the following diagram:

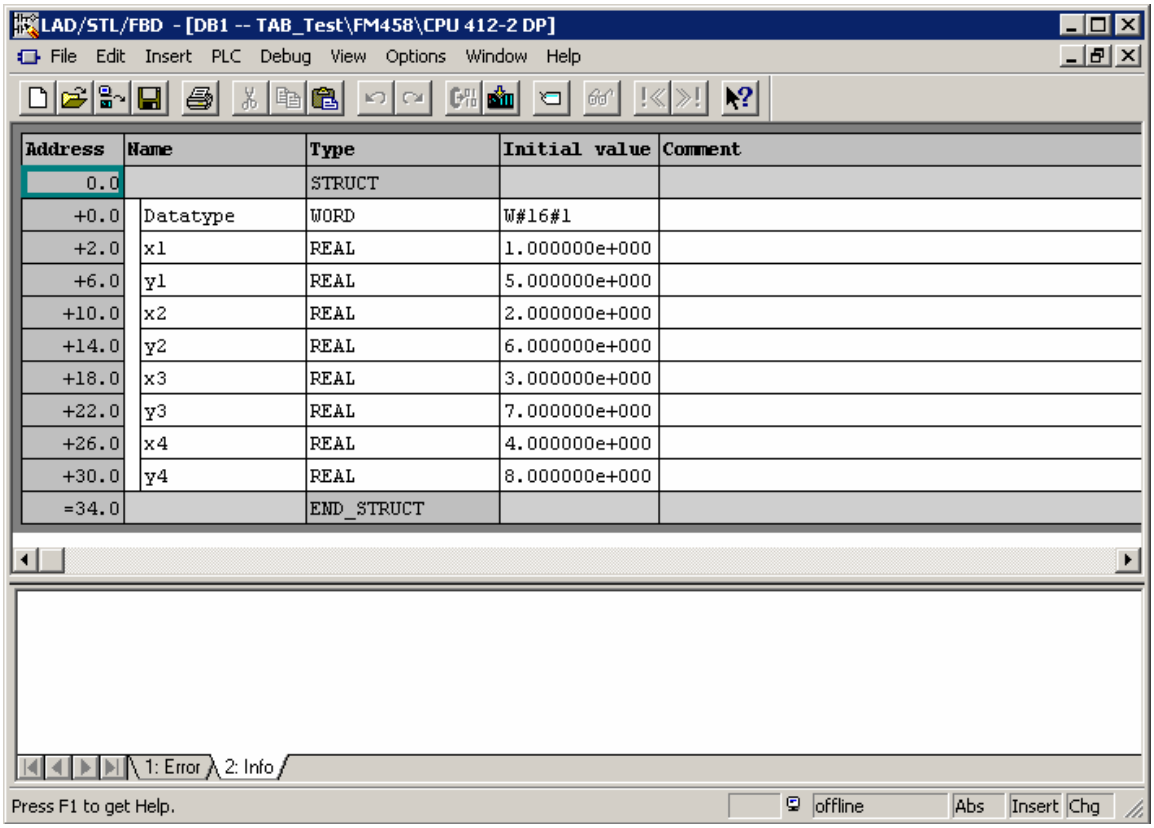


Fig. 3-57 Manually entered tabular values in the "LAD/STL/CSF" application

NOTE

Only values associated with the same data type may be included in a table. For this reason, specifying an ARRAY is an effective way of entering data. This means that the data type doesn't have to be specified each time.
Refer to the online help of the application "LAD/STL/CSF" - especially "Help for STL" for the procedure to make entries for an ARRAY type.

(4) Saving the DBs

After the tabular values have been completely entered, the DB can be saved under "File → Save".
The tabular values are then located in the DB for transfer.

3.10.3.3.2 Importing tabular values

The tabular values, provided in the DB, can also be imported from an external source, e.g. an MS Excel table. However, the following points should be observed for error-free import:

- The source file of the table must have a specific format
- The source file must be linked-in as external source file under STEP7
- A new DB is generated from the external source file
- The necessary points and steps, required for the import operation, will now be explained.

Table format

In order to import an existing table (e.g. generated using Excel) into the DB, it must be compliant with a specific format syntax:

- The table must contain a header, which contains information about the name of the DB and the version.
- Information about the structure and the data type of the tabular values should then be specified.
- The tabular values are then specified (as initial values).
- It should be observed that X and Y values must always be specified, alternating.
- The table should be saved with the *.AWL extension.
- The table can then be used as external source file.

NOTE

The *initial value* is any value which can be defined for each tabular value. It is only used if an actual value is not specified for the associated tabular value.
The tabular values are exclusively defined as *initial values*. *Actual values* are not used.
This significantly reduces the file size and in turn, the required memory.

An example of a table with four X and four Y values, data type REAL is shown in the following diagram:

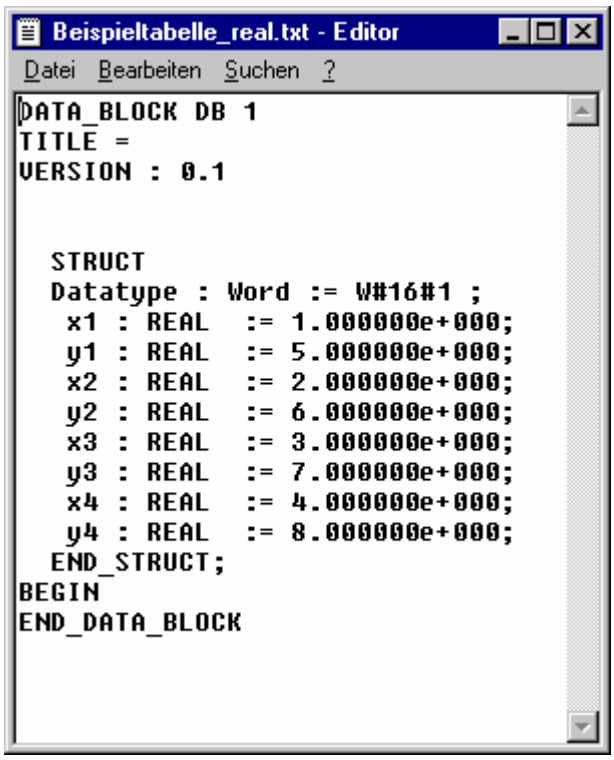


Fig. 3-58 An example of a table with values, data type REAL

An example of a table with two X and two Y values, data type DINT is shown in the following diagram:

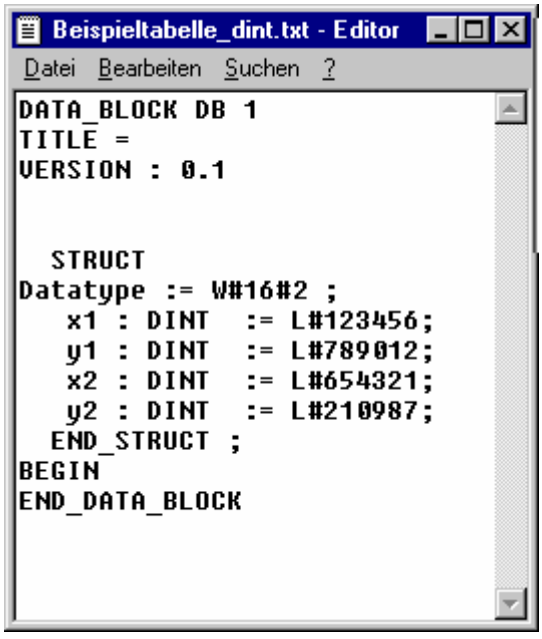


Fig. 3-59 An example of a table with values, data type DINT

From Excel to STL

The following sections explain, using examples, how to re-format an Excel table to obtain the required table format.

The file example, shown in the following diagram, is formatted step-by-step corresponding to the specifications of the required table format.

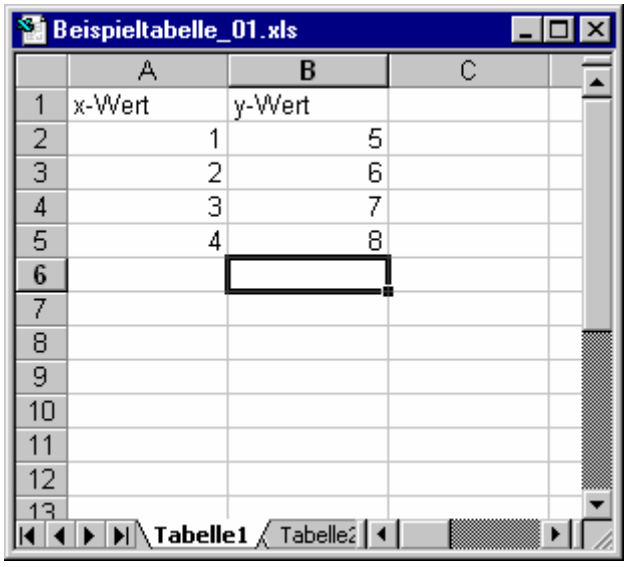


Fig. 3-60 An example of a table in MS Excel

(1) Header

Initially, the required header is inserted. To do this, 5 lines are inserted at the beginning and the following data is entered:

- DATA_BLOCK DB 1 [number of the DB]
- TITLE = [enter as required]
- VERSION : 0.1 [version data]

The Excel table with inserted header is shown in the following diagram:

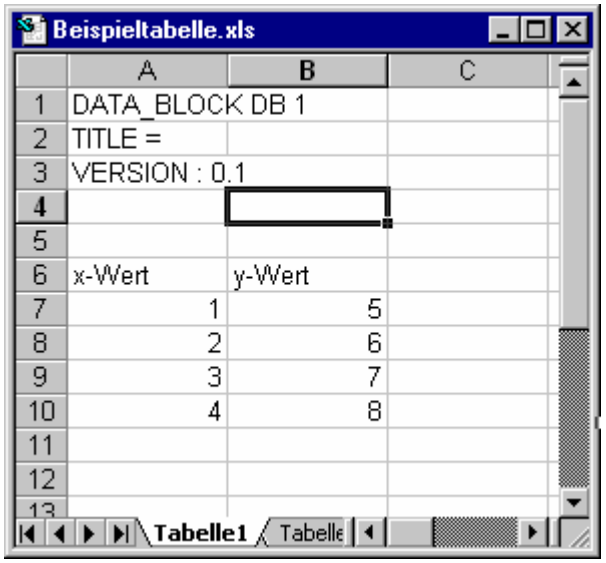


Fig. 3-61 An example of a table in MS Excel with inserted header

(2) Insert structure and tabular values

In a next step, the structure of the tabular values and the values, specifying the data type, are inserted. In this case, two lines plus an initial and end line are inserted for each value pair. Furthermore, a line is inserted at the start to specify the data type used.

The start of the structural data is displayed in the starting line with the "STRUCT" entry. The data type, used in the table, is specified in the following line ("W#16#1" for data type REAL, "W#16#2" for data type DINT).

This is followed by the structural data and tabular values for the individual value pairs, where X and Y values are always entered alternating. The tabular values are specified corresponding to the data type used (in this case REAL). The end of the structural data is displayed in the final line with the "END_STRUCT;" entry.

Finally, only the data for the data section of the actual values has still to be specified ("BEGIN" and "END_DATA_BLOCK"). As the tabular values already have the structural data in the starting (initial) values, it is not necessary to specify the individual actual values.

The Excel table with inserted structural data and tabular values is shown in the following diagram:

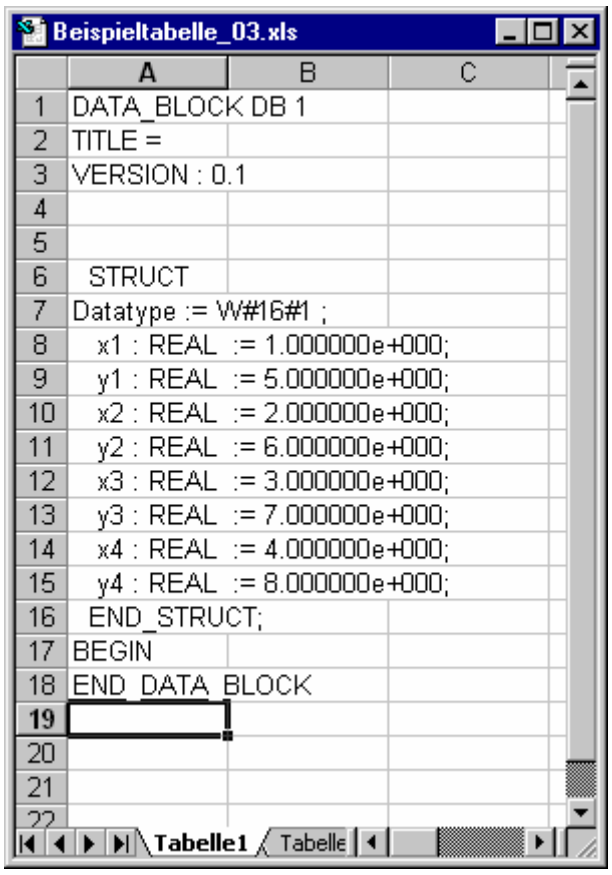


Fig. 3-62 Example of a table in MS Excel with inserted structural data and tabular values

(3) Saving as STL [AWL] file

Finally, the correctly formatted file only has to be saved as text file with the extension *.AWL. In this case, the following should be selected in MS Excel "File → Save as...". "Formatted text (separated by blanks) (*.prn)" file type should be selected and the table example should be saved under a freely selectable name and location.

"Save as" window in MS Excel with the appropriate selection is shown in the following diagram:

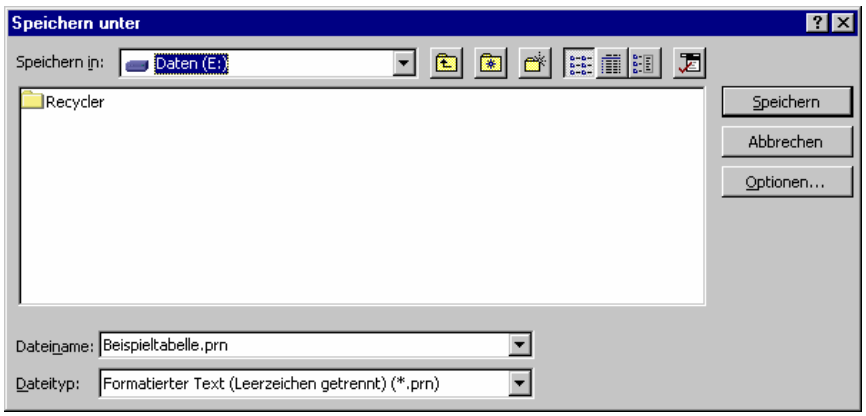


Fig. 3-63 An example of a table in MS Excel saved as text file (.prn)*

After the file has been saved, the file type should be changed from *.prn to *.awl. This file can then be opened with any text editor.

The following diagram shows the table example as STL [AWL] file, opened in the standard Windows text editor:

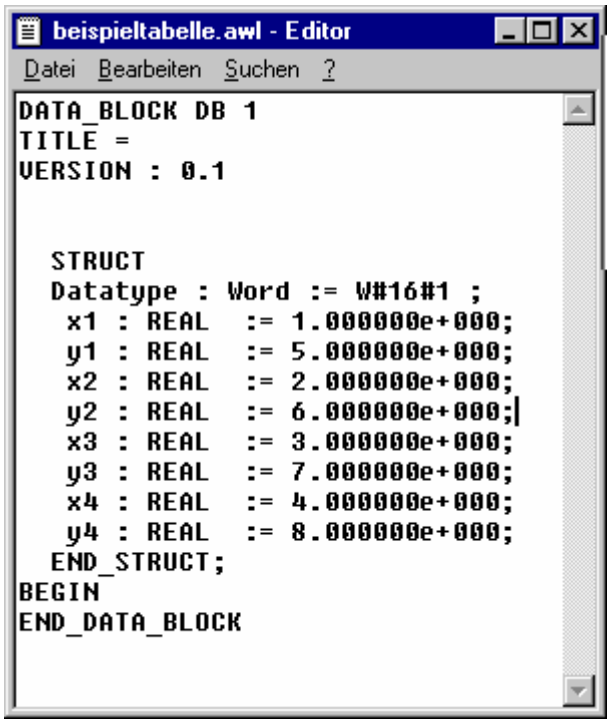


*Fig. 3-64 Table example, saved as *.awl file, opened in the text editor*

This file can only be used as external source file in STEP7 for a DB.

Incorporating the table as source file

Using the file example "BEISPIELTABELLE.AWL", generated above, the individual steps to incorporate an externally generated table in a DB will now be explained.

NOTE

In addition to specifying the tabular values, it is especially important to specify the name of the DB. A DB is subsequently generated using the name specified in the file.
In the above file example, "DB1" is specified as DB name in the first line. (refer to Fig. 10)

Now, an external source is inserted in the STEP7 configured software in the S7 program under "Sources". After selecting "Sources", the context-sensitive menu can be called-up by clicking in the righthand partial window with the righthand mouse key. An external source should be inserted here as new object.

The procedure is shown in the following diagram:

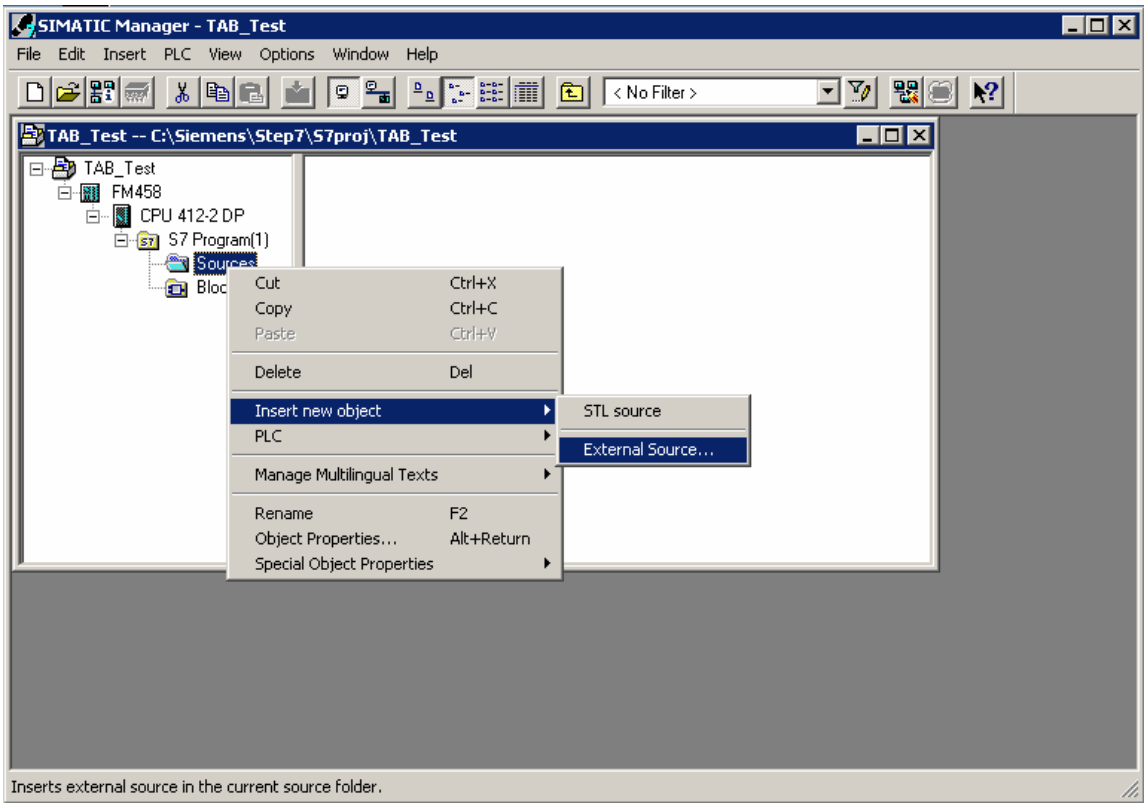


Fig. 3-65 Inserting an external source in STEP7

The STL [AWL] file, generated above, is selected as source file. The following diagram shows the file selection window:

Fig. 3-66 Selecting the file to be inserted in STEP7 as external source

The selected file is opened (in this case: BEISPIELTABELLE.AWL). It now exists as source file in the configured software under "Sources". It is selected there and is opened.

The file example, available under "Sources" and its context-sensitive menu is shown in the following diagram:

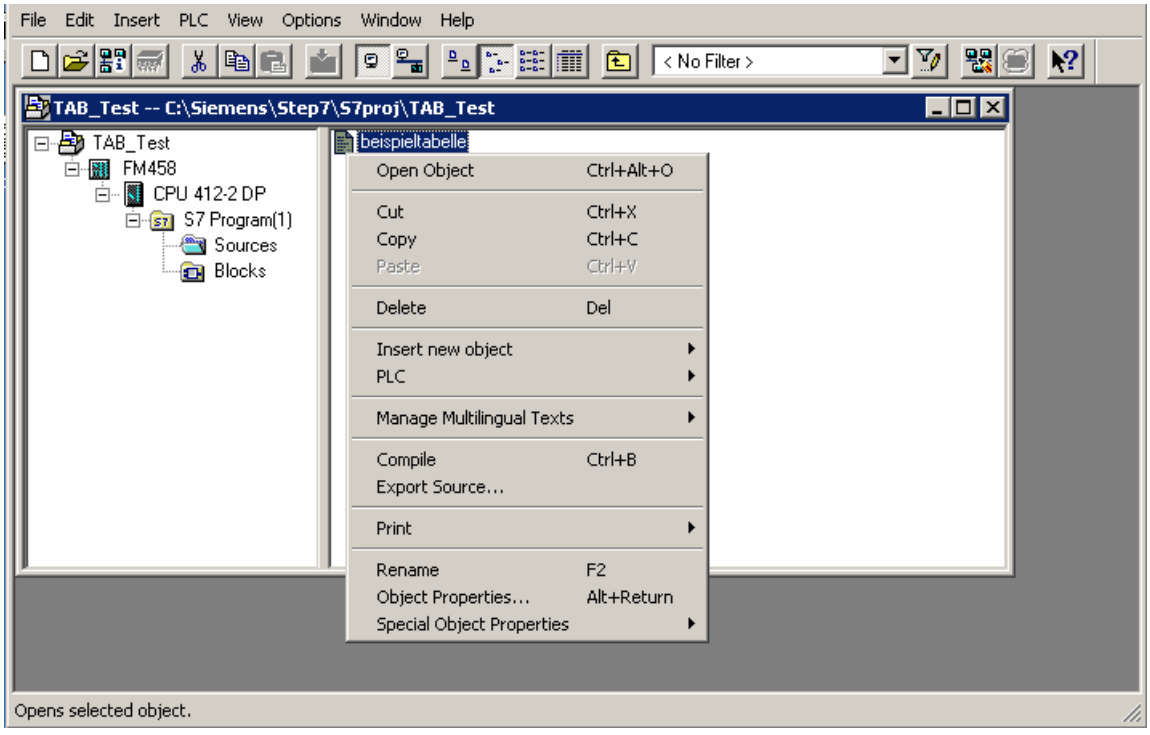


Fig. 3-67 Generated source file in STEP7

After the file has been opened, it can be edited in the "LAD/STL/CSF" program. There it can be compiled via "File / Compile".

The procedure is shown in the following diagram:

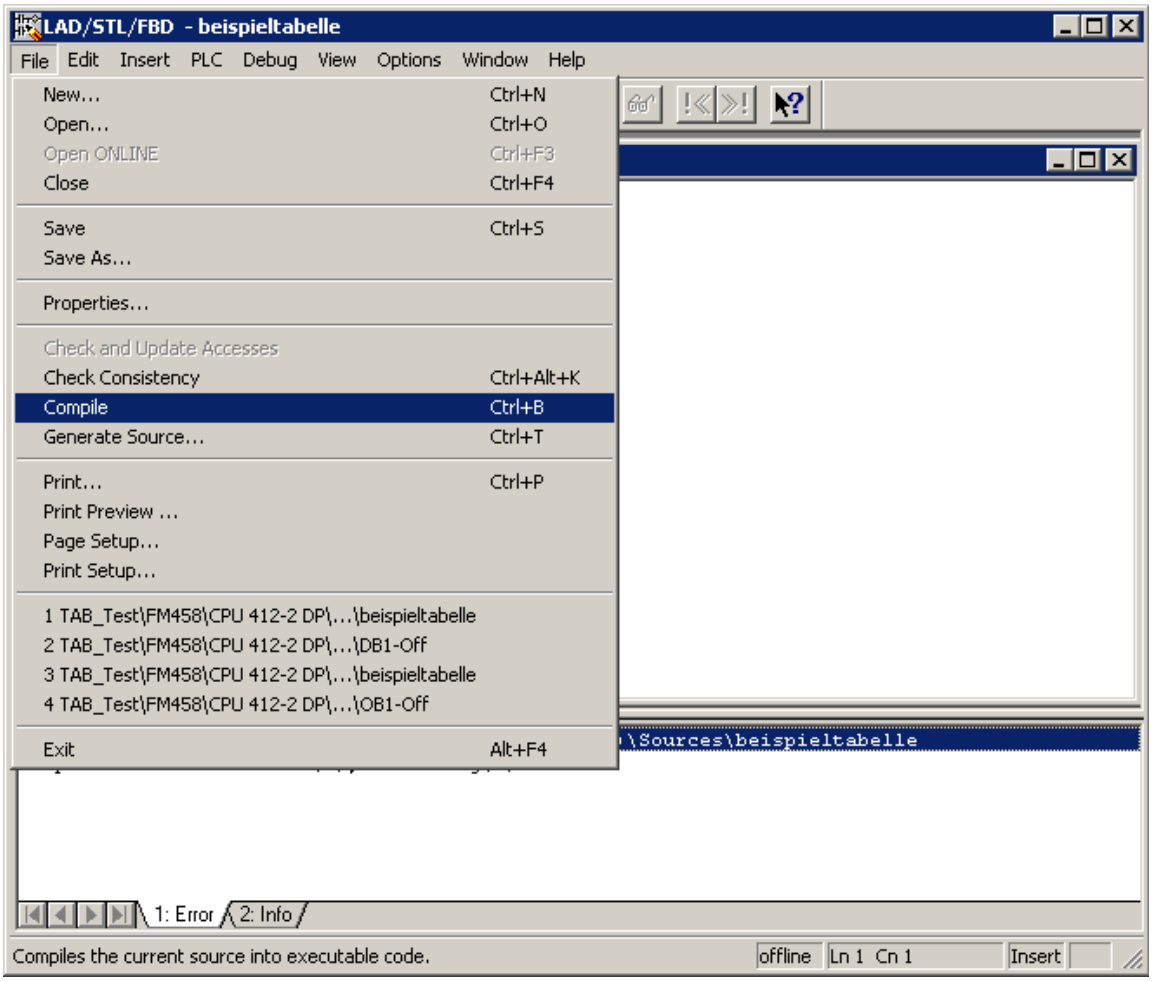


Fig. 3-68 Compiling the source file in the "LAD/STL/CSF" application

After the file has been successfully compiled, a new DB is available in the configured software. The name of the DB corresponds to the name specified in the header line of the file.

The following diagram illustrates the newly generated DB in STEP7 configured software under "Blocks":

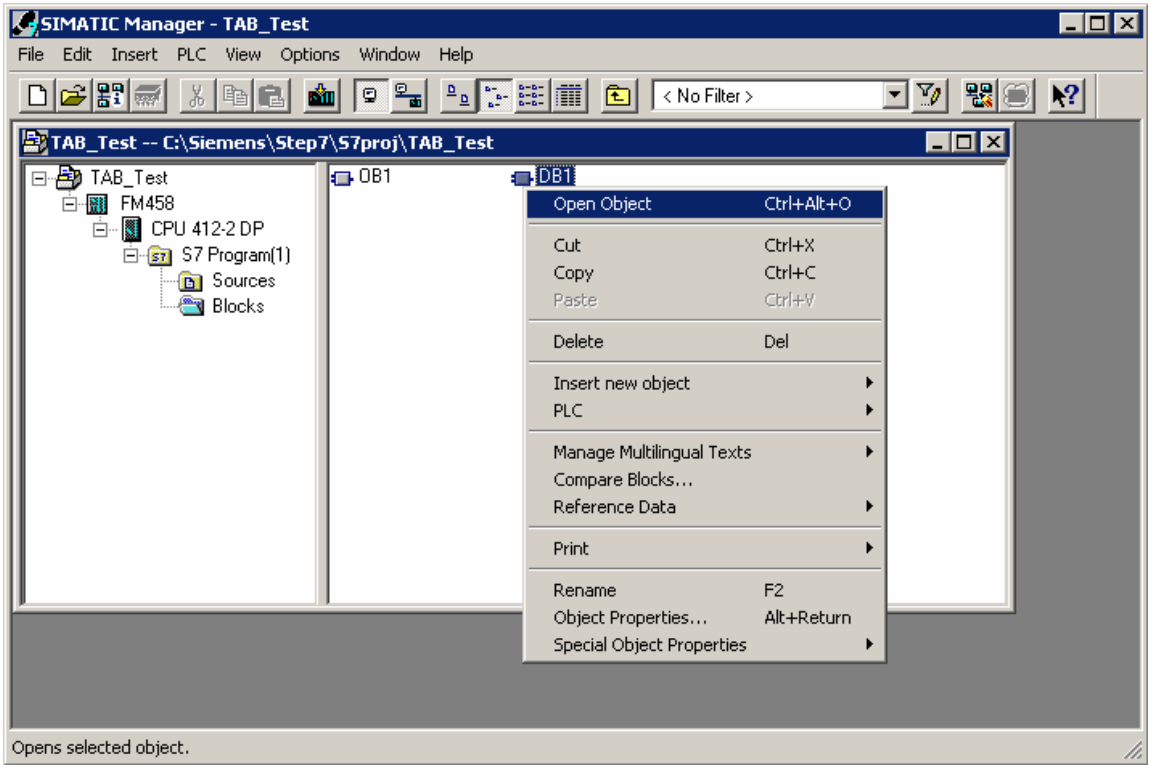


Fig. 3-69 Newly generated DB after compiling the source file

In order to check the contents of the DBs, it can be opened in the "LAD/STL/CSF" program. "Data view" should be selected in the "View" menu to display the initial (starting) values as well as the actual values.

The contents of the opened DB is illustrated in the following diagram:

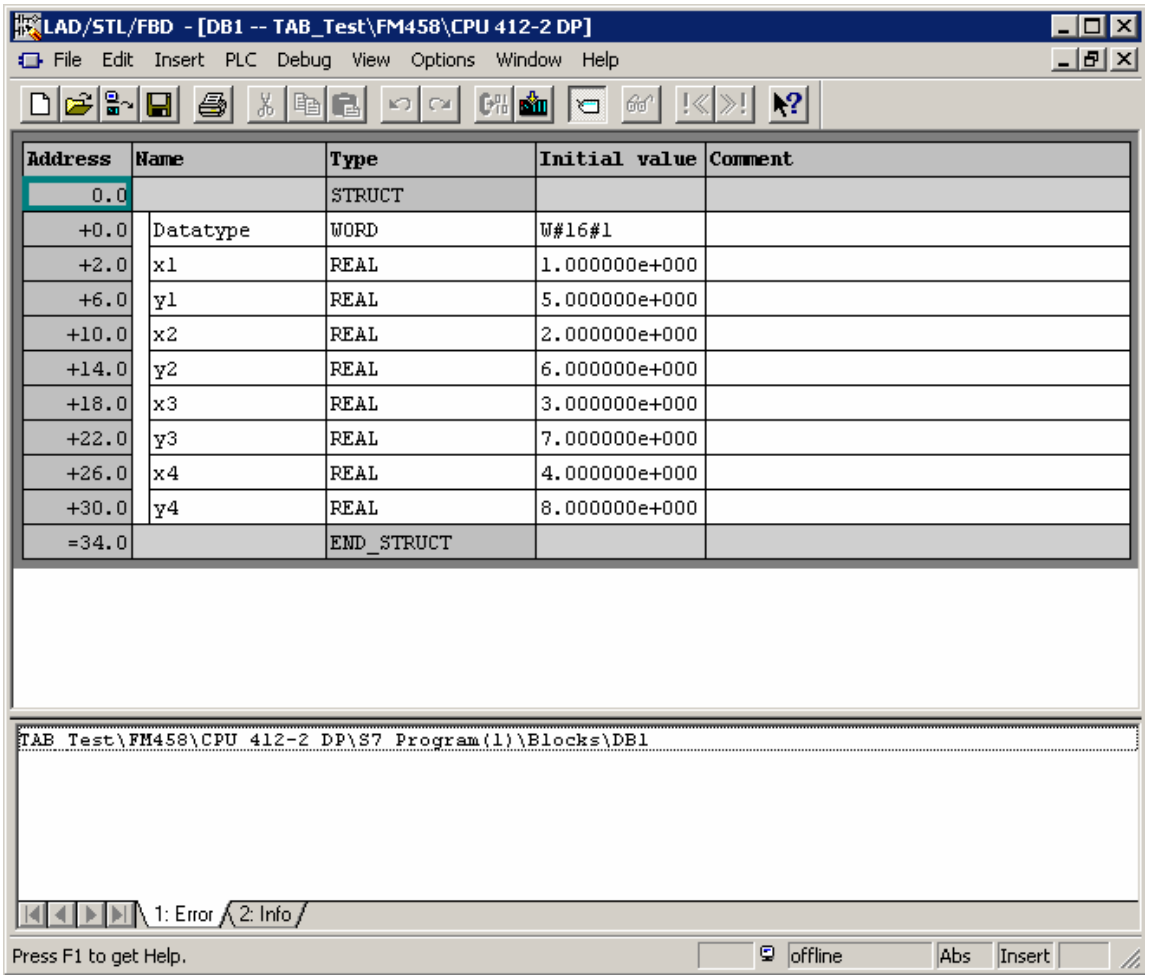


Bild 3-70 Contents of the newly generated DB in the "LAD/STL/CSF" application

3.10.3.3.3 Subsequently downloading tabular values into a DB

If tabular values are to be subsequently downloaded into the DB, because the table is too large and there is not sufficient user memory for several DBs, then the table should be transferred to the SIMATIC FM 458 application module in several sub-sets of the table. To do this, the table must be split-up into sub-sets of the table. The size of the individual sub-set tables should be selected so that the user memory of the S7-CPU is not exceeded. The individual table sub-sets are then transferred one after another.

NOTE It is especially important that the individual table sub-sets are transferred in the sequence of the value pairs. If they are transferred in the incorrect sequence, then the tabular values will not be correctly available in the configured software.

There are two possibilities:

- Manually enter the individual tabular parts at the DB in the "LAD/STL/CSF" application and then transfer this part of the table
- Generate individual source files with different names for each table sub-set and after being successfully linked-into the DB one after the other, then transfer

Manual entry

In order to subsequently download tabular values into a DB manually, the following steps should be carried-out:

- The appropriate DB should be opened by double-clicking in the "LAD/STL/CSF" application.
- The existing tabular values should be replaced by entering the value of the subsequent tabular section.
- The DB should be saved.
- The values of the table sub-sets can now be transferred.

Generating several source files

The following steps have to be carried-out when subsequently downloading tabular values into a DB by generating several source files:

- The same DB name should be specified in the header of the individual source files (*.AWL).
- The individual files may not exceed the memory size of the DB.
- The file names are best numbered in an increasing sequence.
- The individual files are now linked-in as source files as described above. However, they are still not compiled.
- The first source file is compiled and the tabular values, now available in the DB, transferred.
- The second source file is compiled so that its tabular values are now available in the DB. These are now transferred to the S7 control system.
- Analog to this, the other source files are compiled and transferred one after the other.
- After the last table sub-set has been transferred, the LASTDB connection should be set from 0 to 1. This signals that the table has been transferred.

3.10.3.4 Structure of the data telegram for TCP/IP or DUST1 connection

If the communications link involves a TCP/IP or DUST1 coupling, then the data telegram structure must be carefully observed. This is described in the following. The data telegrams are "generated" using the function blocks CTV and CRV.

The data telegram is defined so that all of the tabular values can be transferred in a data block as well as in several data blocks.

The structure of a data block is shown in the following table:

Data type	Description
char [4]	Telegram ID Each table telegram is identified with the "TAB0" ID
u_int16	Telegram commands (bit-coded) 1: New table (rising edge, from 0 -> 1) 2: End of table
u_int16	Data format (REAL=1, DINT=2)
u_int32	No. of the actual data block
u_int32	No. of tabular values (X and Y values) The number of values must always be an even number. This means that always the same number of X and Y values are transferred.
u_int32 [56] / float [56]	Array with tabular values. (X and Y values, always alternating)

The TAB or the TAB_D sends an acknowledgement to the sender for each data block received.

The structure of the acknowledge telegram is shown in the following table:

Data type	Description
char [4]	Telegram ID Identifies each table telegram with the "TAB0" ID
u_int32	No. of the actual data block
u_int32	Status / error numbers 0xB210 OK (data block is o.k.) .....

NOTE New table data is now transferred into the inactive table if the "New table" command is set.
After the "End of table" command has been received, all additional table data are rejected until the "New table" command is received.

3.10.4 Automatic mode: Memory card

Table values can be combined to form components using the D7-SYS additionalComponentBuilder (this is included in D7-SYS V5.2 plus SP1). These components can be downloaded as additional objects on the memory card. From there, they are read-out using the TAB or TAB_D function blocks.

One or several table files are imported in the D7-SYS additionalComponentBuilder, which then combines these files to form a component file (download file), which can then be downloaded onto the memory card.

The D7-SYS additionalComponentBuilder (aCB) does not check the contents of the files. The tables are an exception to this rule. The contents of these table files are checked. If the table file has an erroneous structure, then aCB immediately flags this.

The procedure from generating a table file up to configuring the function blocks is explained in the following sections using an example.

3.10.4.1 Generating a table file in the csv format

The table values are generated as required using a table calculation program (e.g. Excel).

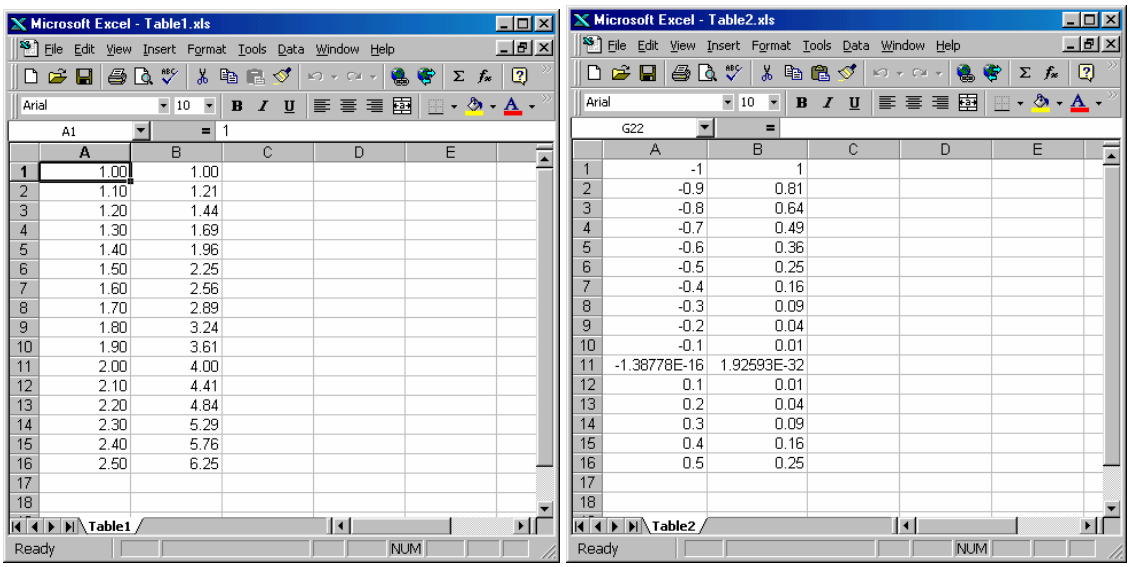


Fig. 3-71 Tables values in Excel

Conditions

The table files must fulfill the following conditions:

- A table file may only comprise two columns – if additional columns are included in the table, an error message is displayed in a dialog window.
- Both of the columns must contain the same number of values. If this is not the case, then the D7-SYS additionalComponentBuilder displays an error message in a dialog window and the table values are rejected.

The D7-SYS additionalComponentBuilder expects the following data format:

- [+/-] xxx.yyy – real value, decimal places are specified using a „.“ (e.g. 145.123)
- [+/-] xxx,yyy – real value, decimal places are specified using a „,“ (e.g. 145,122)
- [+/-] xxx.yyyE+/-mm – real values shown as an exponent, decimal places are specified using a „.“ (e.g. 145.122E+12)
- [+/-] xxx,yyyE+/-mm – real values shown as an exponent, decimal places are specified using a „,“ (e.g. 187,122E+12)

For the „Table DINT“ type description:

- [+/-]xxx – Integer or double integer (e.g. 145)

The following conditions still apply for the table files:

- ASCII files
- The table columns are separated using a semicolon or tab character
- Lines are separated using a line break or semicolon

Saving tables

Tables, which are generated using MS Excel and are saved in the *.csv format or as "Text (Tabs separate)“ fulfill these conditions.

The following diagram shows two example files with table values which were saved in the csv format:

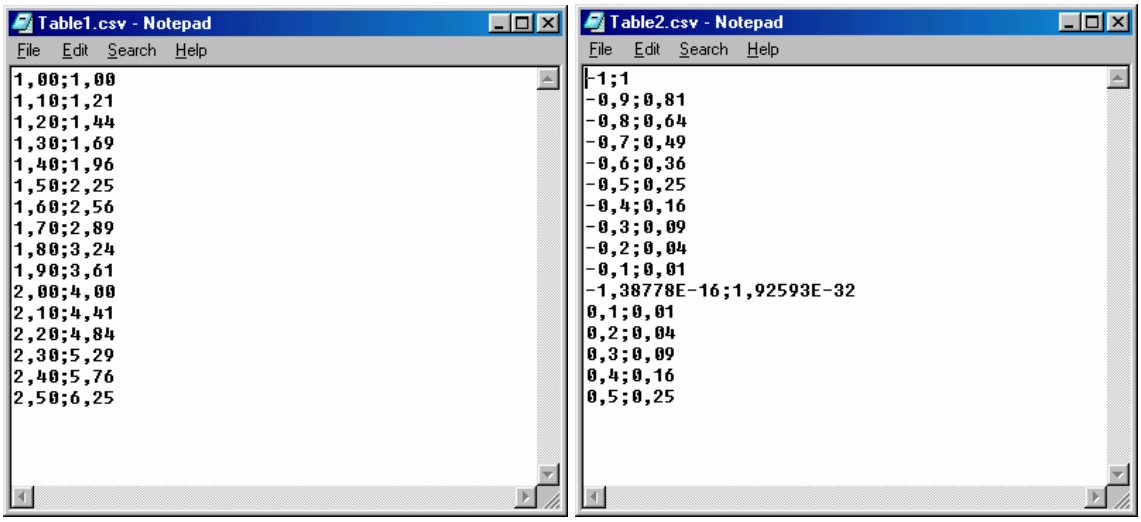


Fig. 3-72 Table values which were separated using semicolons (.csv format)*

3.10.4.2 Working with the D7-SYS additionalComponentBuilder

After the table files were saved in the csv format, they can be imported in the D7-SYS additionalComponentBuilder.

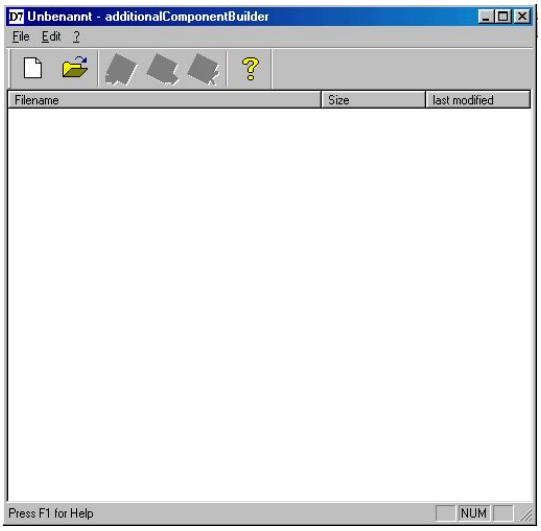


Fig. 3-73 D7-SYS additionalComponentBuilder

In the next step, a new component file is set-up with  . To start, the properties are specified in the following dialog field.

New component

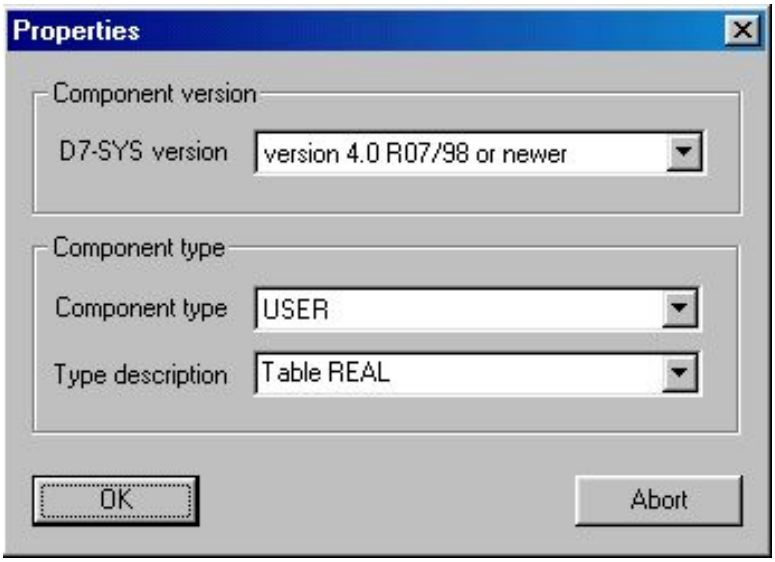


Fig. 3-74 Setting the properties

The following settings should be made:

These properties cannot be changed at a later time and have a gray background.

- **D7-SYS version**
  List box, in which the version is specified for which the components should be generated

- **Component type**
  List box with the fixed entries "USER“, "IT1“ and "IT2“. “USER” is the default value

  The entries have the following significance:

  – USER = Component file generated by the user,
    e.g. table files

  – IT1/IT2 = System component file for ITSP modules

- **Type description**
  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 in the DINT format.

  The entries have the following significance:

  – REAL table: Table file with REAL data type

  – DINT table: Table file with double integer data type

  A new type description can be entered in the list box and acknowledged using RETURN. This new type description is then transferred into the list box and can be selected from the list box the next time.

Saving

The new component file can be set-up after the settings have been completed.
The new component file is, as standard, set-up in C:\temp. If another memory path is specified, then when the program re-starts, this is used as standard memory path.

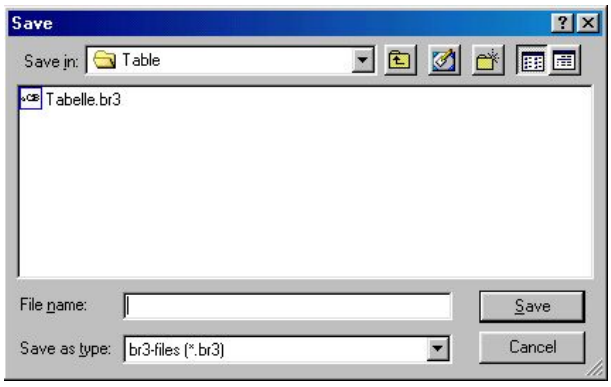


Fig. 3-75 Saving the new component file

Table files can now be added. A file selection window is opened using  with which the required table files can be selected.

NOTE

Only tables with a uniform value format can be included in a component with the "table" type description! This means that a REAL table only contains tables with REAL values.

The following diagram shows the contents of the D7-SYS additionalComponentBuilder after importing the two generated table files:

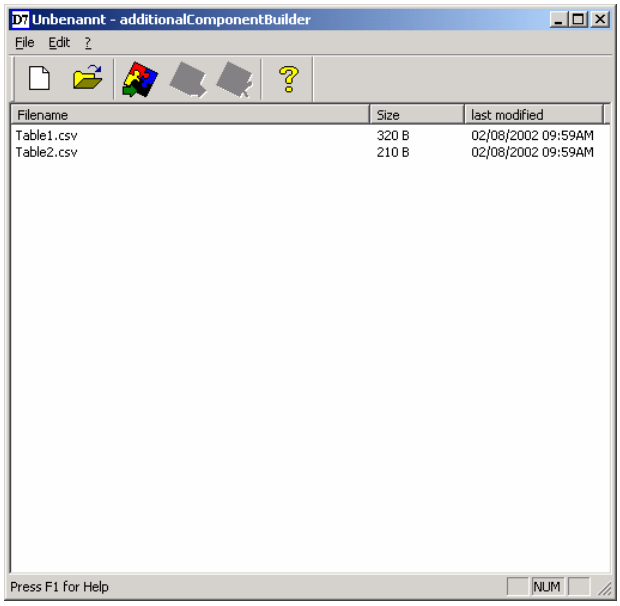


Fig. 3-76 D7-SYS additionalComponentBuilder with imported table files

Additional table files can be added or imported or deleted at any time. The D7-SYS additionalComponentBuilder automatically takes-over the management of the table files and saves the modified component files.

Opening

When opening existing components, "C:\temp“ is the standard search path of the D7-SYS additionalComponentBuilder. If another path is selected, when the program re-starts, this is used a standard search path.

3.10.4.3 Downloading

After the component file was set-up with the D7-SYS additionalComponentBuilder, it can be downloaded into the general download dialog box.

(1) Opening the download dialog box in D7-SYS with “target system→ Download”
Using this dialog box, the current configuring can download the optional components into a memory card (offline/online).

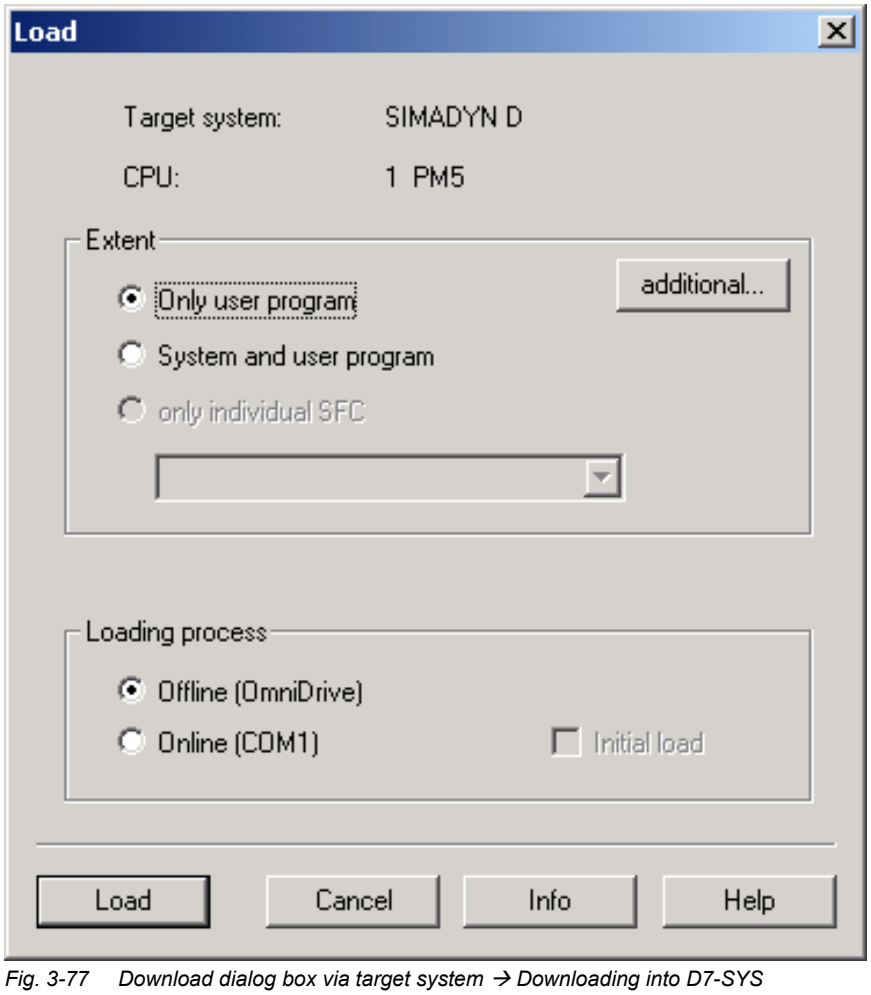


Fig. 3-77 Download dialog box via target system → Downloading into D7-SYS

(2) Opening the dialog box for optional components
A maximum of 2 components can be selected. A file can be selected for the selected components by clicking on the “NEW” button.

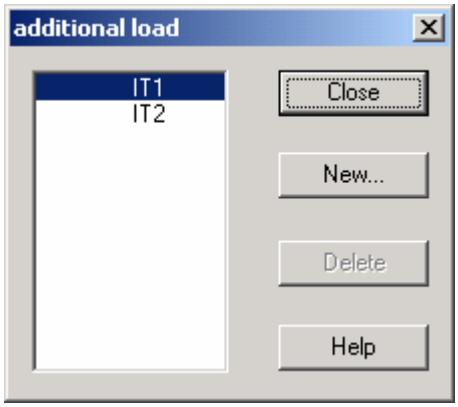


Fig. 3-78 Selection dialog box for optional components, e.g. table data

(3) A file selection dialog box opens to select additional components

The component file, previously created using the D7-SYS additionalComponentBuilder, is now assigned the component IT1 and during the next download operation, is written into the memory card.

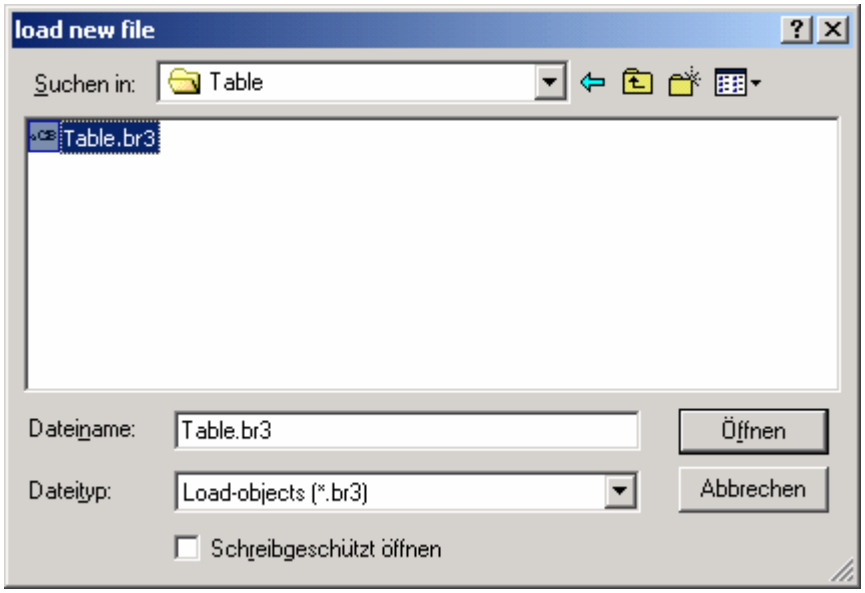


Fig. 3-79 Downloading a component file

3.10.4.4 Configuring the function blocks

For the "automatic mode, memory card" mode, only the TAB and/or TAB_D function blocks must be configured, depending as to whether table values, REAL data type and/or DINT data type have to be managed. Each table may only contain values of one data type. If several tables are to manage various data types, then a TAB or TAB_D should be configured for each table.

The TAB and TAB_D function blocks should be configured in a sampling time greater than or equal to 32ms. The following connection settings are required:

CTS= 0

US = Not assigned

NAM = Name of the table file (with file name extension which was defined when "saving", e.g. MS Excel)

AUT = 1 (automatic mode activated)

The configuring is shown in the following diagram:

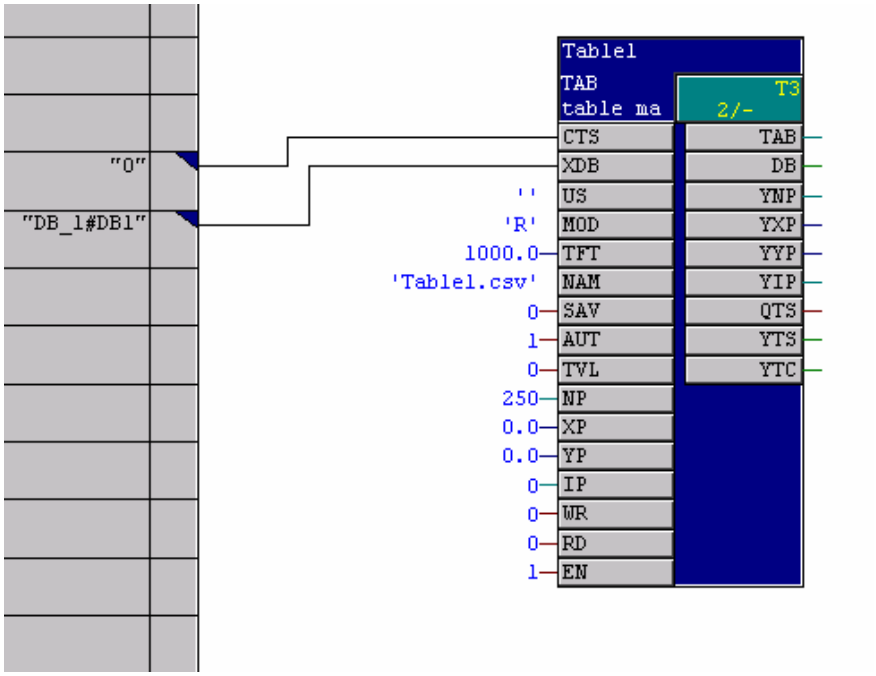


Fig. 3-80 Configuring the TAB function block

The table function blocks for 2 tables are shown in the following diagram. The table values, which are now managed by the function blocks, can now also be used by additional function blocks, e.g. FB TABCAM.

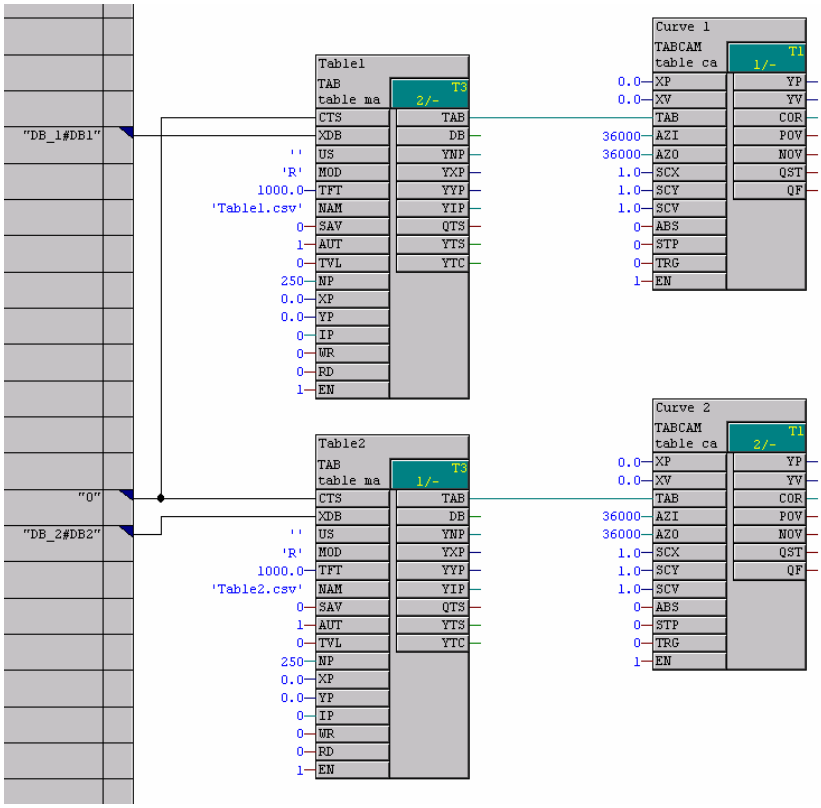


Fig. 3-81 Configuring example

3.11 Communications utility, message system

General

The message system allows the user to log certain events which he has selected. A description of these events is collected in the message sequence buffer and is then made available to the user via a data interface.

Configuring

The message system operates purely 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 the message system

The message system consists of 3 types of function blocks:

- **Central block @MSC**
  The central block sets-up the required data structures and administers them. It is also responsible in evaluating communication- and system error messages.
- **Message entry blocks MER ...**
  Message entry blocks generate messages when an input changes. Message entry blocks can mutually interrupt each other. Thus, the messages do not have to be entered in the message sequence buffer in the sequence in which they occurred.
  The message entry blocks differ by:
  - the number of messages which can be generated.
  - the capability of being able to process additional incoming process conditions/statuses in the form of measured values.
- **Message evaluation blocks MSI ...**
  Message evaluation blocks output messages, generated by the message entry blocks, via a data interface and make them accessible to the user.

3.11.1 Entry logic of the message entry blocks

3.11.1.1 Message entry blocks for an activated message

Entry logic

For message entry blocks, which only generate an activated message, the following conditions must be fulfilled for message entry:

- input EN must be set.
- a positive edge must be available at input I1.
- connection Q1 or SM must be reset.

If the conditions are fulfilled, a message is generated and connection Q1 is set.

If the conditions are not fulfilled, then, if connection SM is reset, connection Q1 is also reset.

3.11.1.2 Message entry blocks for an activated and a de-activated message

Entry logic

For message entry blocks, which generate an activated and a de-activated message, the following conditions must be fulfilled for message entry:

- input EN must be set.
- for an activated message, a rising edge must be available at input I1 and connection Q1 or SM must be reset.
- for a de-activated message, a falling edge must be available at input I1 and connection Q2 or SM must be reset.

If these conditions are fulfilled, then:

- for a rising edge, an activated message is generated and connection Q1 is set.
- for a falling edge, a de-activated message is generated and connection Q2 is set.If these conditions are not fulfilled, and if connection SM is reset, connections Q1 and Q2 are reset.

Special features for MER16, MERF16, MER0, MERF0

For message entry blocks MER16, MERF16, MER0, MERF0, which have a vector as message connection, and which generate 16 or 32 messages, for message connection IS1 and output connection QS1 or QS2, the appropriate bit positions must fulfill the conditions of the entry logic.
Further, these blocks have a QN output, which indicates whether a message was generated.

3.11.2 Configuring example for a message system

Prerequisites for a message system

- Subrack
- At least one CPU in the subrack
- A data interface is available with the name ”D01”

Function blocks required

In the example, only the actually required blocks for the message system are listed. Central communication blocks (e. g. for the data interface) are not listed.

The configured message system consists of:

- 1 central block @MSC
- 2 entry blocks (MER and MERF0)
- 2 message evaluation blocks (MSI and MSIPRI)

Name and message buffer

The name of the message system is ”MELD”. This name is configured at all CMS connections of the message blocks. The message buffer can accommodate 30 messages (connection NOM at @MSC), is located in a volatile RAM (connection SAV at @MSC) and is enabled for message entries (connection MUN at @MSC).

Assigning message and block

Generated messages can be assigned to blocks using the RP- and RRS connections, whereby each block of the message system has at least one RP connection. Proceed as follows:

- **Prefix 0**
  Designates a message which is generated by @MSC (communications- and system error messages). Thus, connection RP of @MSC is assigned the value 0. @MSC automatically generates the suffix, depending on the message type.

- **Prefix 1**
  Designates a message, which is generated by MSI (overflow messages). Thus, MSI assigns a value of 1 to connection RP. MSI automatically generates the suffix (number of messages which have overflowed).

- **Prefix 2**
  Designates a message which was generated by MSIPRI.

- **Prefix 3**
  Designates a message, which was generated from a message block (MER or MERF0). Thus, connection REP of MER and MERF0 is assigned the value 3. The suffix is not automatically generated as for the other blocks. In this case, the connections are available, at which the suffix can be configured. 33 various messages are generated in the example (1 MER message, 16 activated messages MERF0, 16 de-activated messages MERF0), which are numbered from 0 - 32:

  - The message of block MER is assigned suffix 0 (RS connection MER).
  - The 16 activated messages of block MERF0 are assigned suffix 1-16 (RS1 connection MERF0).
  - The 16 de-activated messages of block MERF0 are assigned suffix 17-32 (RS2 connection MERF0).

- **Suffix**
  For block MERF0, for the suffix a basis value is specified. The bit number of the message-generating bits of message signal vector IS1 is added to this basis value.

Functional combination of the messages

Using a prefix and suffix, it is not only possible to uniquely assign the messages to the generating blocks, but it is also possible to functionally combine the messages. In the configuring example, the MER and MERF0 blocks generate messages with the same prefix, which indicates a logical association.

Channel on the data interface

In the configuring example, both message evaluation blocks set-up a channel at the data interface D01 in the "select" mode (thus, the same channel name can be configured).

Measured value input and message signals

The measured value input of block MER is not connected in the particular example. At the measured value input, a process condition is normally applied. The message signals of function block MERF0 act similarly.

Generating and reading-out messages

Messages are generated by a rising edge at connection I1 of block MER or by a changing value at input IS1 of block MERF0. The message evaluation block immediately reads-out the first message from the message buffer and transfers into the data channel as both blocks are "enabled" (input EN=1). Additional messages are only transferred into the data channel when the previous message has been read-out of the channel.

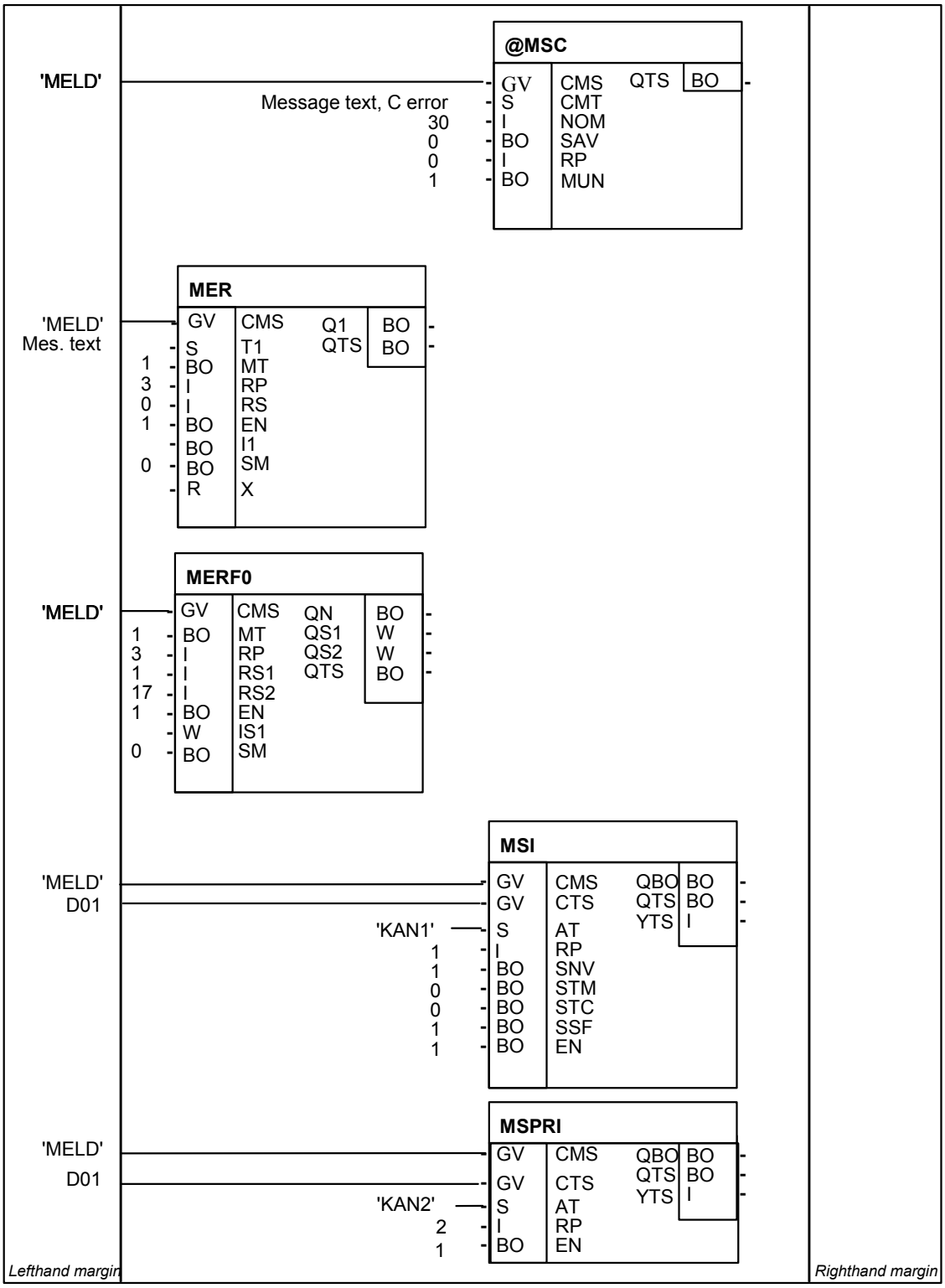


Fig. 3-82 Configuring example, communications utility, message system

3.11.3 Output formats of the message evaluation block MSI

3.11.3.1 Structure of an error- or alarm message

General

The message evaluation block MSI has four inputs to select the format:

- input SNV
- input STM
- input STC
- input SSF

The message format is important for the receiver of a message and its interpretation.

Message text length

Input STC defines the message text length. It is set to a constant length (60 characters) using STC = 1. If a message text is shorter than the maximum length or is not available, it is filled with blanks. The advantage is the constant number of data which is to be transferred. This connection has no effect on the remaining structure of the message and the message type description.

Message text format

Inputs SNV, STM and SSF are evaluated once during the initialization phase and then define the format of the messages output. The messages are output at the channel, specified at input AT at the data interface specified at input CTS.

3.11.3.2 Overview of the message formats

Spontaneous ID

The spontaneous ID has a constant value of 0 and is of no significance.

Sequence number

The sequence number is provided for reasons of reliability and counts the number of messages transmitted so that the receiver can identify which messages have been lost. The sequence number lies in the range from 0-255. When the sequence number has reached the maximum value of 255, when the next message is transmitted, the minimum value 0 is used.

Message type description

Essentially a differentiation is made between the standardized and hexadecimal formats. For a standardized format, the individual values are transferred in the IEEE 754 or ISO 646 standard, which defines a normalized 32-bit floating point notation. The messages, both in the standardized as well as in the hexadecimal format, include a message type description which provides information about the format, selected by the initialization inputs and other parts of the message. The message type description is a bit vector, which should be interpreted as follows:

- Bit 0: If this bit is set, message numbers are output (copy of input SNV).
- Bit 1: If this bit is set, a message text is output (copy of input STM, unless the message is empty).

- Bit 2: If this bit is set, the messages are output in the standardized format, otherwise in the hexadecimal format (copy of input SSF).
- Bit 3: If this bit is set, a measured value is present.
- Bit 4: If this bit is set, then a units text is present. The units text can only be present if there is also a measured value. If there is no measured value or units text, the appropriate message errors are of no significance and are in an undefined condition.
- Bit 5-7: Unassigned

Message type

The message type consists of a character, which specifies the message event type, whereby the following is defined: "S" system error, "C" communications error, "F" error messages and "W" warning messages. The first two message types are only generated by the message system central block.

Message prefix

Corresponds to the input value at RP of the entry block.

Message suffix

Corresponds to the input value at RS of the entry block.

Measured value units and scaling factor

In the hexadecimal format, the measured value description consist of:

- a 32-bit scaling factor which is output in the floating format
- the measured value acquired by the acquisition block
- a measured value data type (SIMADYN D / SIMATIC TDC data type as ASCII character sequence)
- an 8-character measured value unit

HEX format and standardized format

As the precise data format must be specified in the hexadecimal format when initializing data transfer, and on the other hand, the measured value can vary in the size of the notation (0,2 or 4 bytes), for measured values, 4 bytes are always transferred. If the measured value occupies less than 4 bytes, which can be recognized at the measured value data type, then the subsequent bytes cannot be assigned.

In the standardized format, only the scaled measured value and the 8-character long measured value units are transferred.

Message instant

The message instant is transferred in the hexadecimal format in the MMS format, time and date (reference point 1.1.84).
In the standardized format, the message instant is transferred as ASCII character sequence, which includes date (day, month, year) and time of day (hours, minutes, seconds, milliseconds). Date and time of day are separated by a hyphen. The character sequence is 24 characters long (example: ”01.05.1993 08:01:15:0045”).

Message text

The message text is always transferred as ASCII character sequence. In this case, length information is not transferred. This is calculated from the total number of data received. The message text can be a minimum of 60 characters long.

3.11.3.3 Structure of an overflow message

Overflow message

If the message sequence buffer overflows, then the MSI/MSPRE generates an overflow message:

- The overflow message is the warning type (’W’).
- The prefix includes the value at input RP of function block MSI which generates the message.
- The suffix includes the number of messages which have been lost.
- There is no measured value. This is indicated in the message type description.
- The time, at which the message evaluation block generated the overflow message, is entered as message instant.
- The ”sequence buffer overflow” text is output as message text if input STM of the function block MSI is set.

3.11.3.4 Structure of a communications error message

Communications error message

The central block evaluates the communication errors occurring in the system and generates the following communication error messages:

- A communications error message is message type C error (’C’).
- The prefix includes the value at input RP of the central block which generated the message.
- The suffix includes the error number of the C-error message (this is always positive).
- If a measured value is not available, then this is indicated in the message type description.
- The text, configured at input CMT at the central block is output as message text, if input STM of function block MSI is set.
- If the communications error field has overflowed, after all of the C-error messages have been output, a message is generated which includes, as suffix, the negative number of the messages which have been lost. After this message, MSI does not output any additional C-error messages. The instant at which the central block identified the communications error field overflow, is entered as message instant.

3.11.3.5 System error message structure

System error message

A system error message has essentially the same structure as a communications error message. The only differences are the "message text" where the "system message" is always used, as well as the message type ('S'). Further, a maximum of one system error message is generated, which is identified during the initialization phase of the central block.

As suffix, an ID is entered by the system error, which has the following significance:

Value, suffix	Significance
1	5 V power failure
2	15 V power failure
3	Software processing faulted
4	Error when accessing the L-bus communications buffer memory
5	Error when accessing the C-bus communications buffer memory
6	Error when accessing the standard periphery
7	Error when accessing the special periphery
8	Undefined L-bus access
9	Undefined C-bus access
10	(not used)
11	Hardware fault which cannot be identified
12	(not used)
13	Fault/error which cannot be identified
14	Fault message (ready internal) from the local expansion bus (LE bus)
15	Error when accessing the local periphery (LP bus)
16	Overrun of the system bus controller

Table 3-28 Suffix, system error message

3.11.3.6 Detailed description of the message formats of function block MSI

General

The description of the message formats consists of 3 parts:

- Assigning initialization inputs SNV, STM and SSF
- Basic format and maximum length of the message. This length corresponds to the size of the channel logged-on by MSI.
- Net data structure which is required to initialize the channel.
- Input STC connection is not listed here. For STC = 1, the length specification for the message text always corresponds with the maximum length; for STC = 0, it corresponds to the actual message text length.

SNV=TRUE (message numbers available) STM=TRUE (message text available) SSF=TRUE (standardized format)				
Contents	**Message structure (max. 108 bytes)**	**Net data structure**	**Data format**	**No. of data**
Spontaneous ID	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Message type descrip.	1 Octet	2. variable unit	Octet-String	2
Message 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
Message instant	24 characters			
Message text	max. 60 characters			

Table 3-29 Standard format with number and text

SNV=FALSE (message numbers not available) STM=TRUE (message text available) SSF=TRUE (standardized format)				
Contents	**Message structure (max. 100 bytes)**	**Net data structure**	**Data format**	**No. of data**
Spontaneous ID	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Message type descrip.	1 Octet	2. variable unit	Octet-String	2
Message 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
Message instant	24 characters			
Message text	max. 60 characters			

Table 3-30 Standard format without number with text

SNV=TRUE (message numbers available) STM=FALSE (message text not available) SSF=TRUE (standardized format)				
Contents	**Message structure (max. 48 bytes)**	**Net data structure**	**Data format**	**No. of data**
Spontaneous ID	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Message type descrip.	1 Octet	2. variable unit	Octet-String	2
Message 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
Message instant	24 characters			

Table 3-31 Standard format with number without text

SNV=FALSE (message numbers not available) STM=FALSE (message text not available) SSF=TRUE (standardized format)				
Contents	**Message structure (max. 48 bytes)**	**Net data structure**	**Data format**	**No. of data**
Spontaneous ID	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Message type descrip.	1 Octet	2. variable unit	Octet-String	2
Message 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
Message instant	24 characters			

Table 3-32 Standard format without number and text

SNV=TRUE (message numbers available) STM=TRUE (message text available) SSF=FALSE (HEX format)				
Contents	**Message structure (max. 92 bytes)**	**Net data structure**	**Data format**	**No. of data**
Spontaneous ID	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Message type descrip.	1 Octet	2. variable unit	Octet-String	2
Message 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
Message instant	Time and date	7. variable unit	Time and Date	1
Message text	max. 60 characters	8. variable unit	Visible-String	60

Table 3-33 Hexadecimal format with number and text

SNV=FALSE (message numbers not available) STM=TRUE (message text available) SSF=FALSE (HEX format)				
Contents	**Message structure (max. 88 bytes)**	**Net data structure**	**Data format**	**No. of data**
Spontaneous ID	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Message type descrip.	1 Octet	2. variable unit	Octet-String	2
Message 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
Message instant	Time and date	6. variable unit	Time and Date	1
Message text	max. 60 characters	7. variable unit	Visible-String	60

Table 3-34 Hexadecimal text without number with text

SNV=TRUE (message numbers available) STM=FALSE (message text not available) SSF=FALSE (HEX format)				
Contents	**Message structure (max. 32 bytes)**	**Net data structure**	**Data format**	**No. of data**
Spontaneous ID	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Message type descrip.	1 Octet	2. variable unit	Octet-String	2
Message 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
Message instant	Time and date	7. variable unit	Time and Date	1

Table 3-35 Hexadecimal format with number without text

SNV=FALSE (message numbers available) STM=FALSE (message text not available) SSF=FALSE (HEX format)				
Contents	**Message structure (max. 28 bytes)**	**Net data structure**	**Data format**	**No. of data**
Spontaneous ID	Unsigned8	1. variable unit	Unsigned8	2
Sequence number	Unsigned8			
Message type descrip.	1 Octet	2. variable unit	Octet-String	2
Message 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
Message instant	Time and Date	6. variable unit	Time and Date	1

Table 3-36 Hexadecimal format without number and text

3.11.3.7 Output format of the message evaluation block MSIPRI

General

Contrary to the message evaluation block MSI, the format of the messages of the MSIPRI evaluation block can be freely selected. Here, only one format is output. Thus, there are no connections to select a format when configuring the block. The MSIPRI block has been especially developed to output messages on a printer. All of the messages are output as text and with line feed. A message consists of a maximum of two lines.

Structure of the 1st line

Character of the 1st 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	Message type (’C’,’F’,’W’ or ’S’)	One character
47-50	Text: ”Nr:”	
51-53	Sequence number	Max. 3 characters and right justified
54	Text: ” ”	
55-67	Measured value (optional: this is only entered if the message contains a measured value)	Is output as floating value in the following sequence: • sign (positive = "+", negative = "-") • number of places before the decimal point followed by a decimal point and 6 places after the decimal point • exponent, started with the character ’e’ • sign (positive = "+", negative = "-") as well as 2 exponent positions
68	Blanks (optional)	
69-76	Measured value unit (optional: is only entered if the message contains a measured value)	8 characters
77, 78	Special characters, CR and LF	Line feed

Table 3-37 Structure of the MSIPRI evaluation block message, 1st line

Structure of the 2nd line

The second line contains the message text, and is only output if there is a message text. Otherwise this is completely eliminated.

Character of the 2nd line	Significance	Output format
1-60	Measured value text (optional)	Variable length
61, 62	Special characters, CR and LF	Line feed

Table 3-38 Structure of the MSIPRI evaluation block message, 2nd line

Example of a message output

```
”01.05.1993 08:01:15:0045 P:  123 S:   10 Typ: W Nr: 25 -1.123456e+12 ms   ”
”This is a message text”
```

Table 3-39 Example of a message output

NOTE Overflow-, communication error- and system error messages have the same logical structure as for block MSI.

3.12 Communications utility process data

Application

The communications utility, process data supports "pure" data transfer in the transmit- and receive directions, i.e. the function blocks only transfer process data. The data itself is neither evaluated nor logically interpreted.

There are two block classes for data transfer:

- receive- and transmit blocks: CTV and CRV
- channel marshalling blocks: CCC4 and CDC4

The CRV and CTV blocks can handle most of the communication applications.

3.12.1 Receive- and transmit blocks

General

There is one receive- and one transmit block. They are called CRV (communication receive virtual) and CTV (communication transmit virtual).

Using a receive- or transmit block a telegram is configured, which is transferred from or to a coupling module. The structure and contents of the telegram are defined when configuring the virtual connections.

3.12.1.1 Virtual connections

General

A virtual connection is an "invisible" connection between block connections. There is no interconnection drawn at the configuring interface, and only a margin connection is created.

The configuring engineer defines which values are to be transferred from block outputs or to block inputs. He does this using "connection name receive/transmit" at the receivers or transmitters and the "virtual connection name" and the "sequence number" at the block inputs or block outputs to be processed.

Connection name

The connection name consists of an exclamation mark ("!") and a maximum of 6 characters (upper case letters or numbers). The character sequence is located directly after the exclamation mark (e.g. "!SEND"). The exclamation mark does not have to be configured as it is automatically generated.

A virtual connection consists of:

- virtual connection name
- sequence number

Connection name and sequence number are separated by a point (e.g. "!SEND.0056"; the point (period separator) between the connection name and the sequence number does not have to be configured as it is automatically generated).

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, data types STRING (S) or GLOBAL VARIABLE (GV).

Telegram structure The virtual connections with the same connection name (data) define a telegram with a specific structure. The sequence of the data within the telegram is defined by the sequence number. The data with the lowest number is located at the start of the telegram; that with the highest number, at the end. The sequence number defines the relative position of the data in the telegram. Gaps in the sequence numbers are ignored.

Configuring example

Receiving and transmitting with virtual connections.

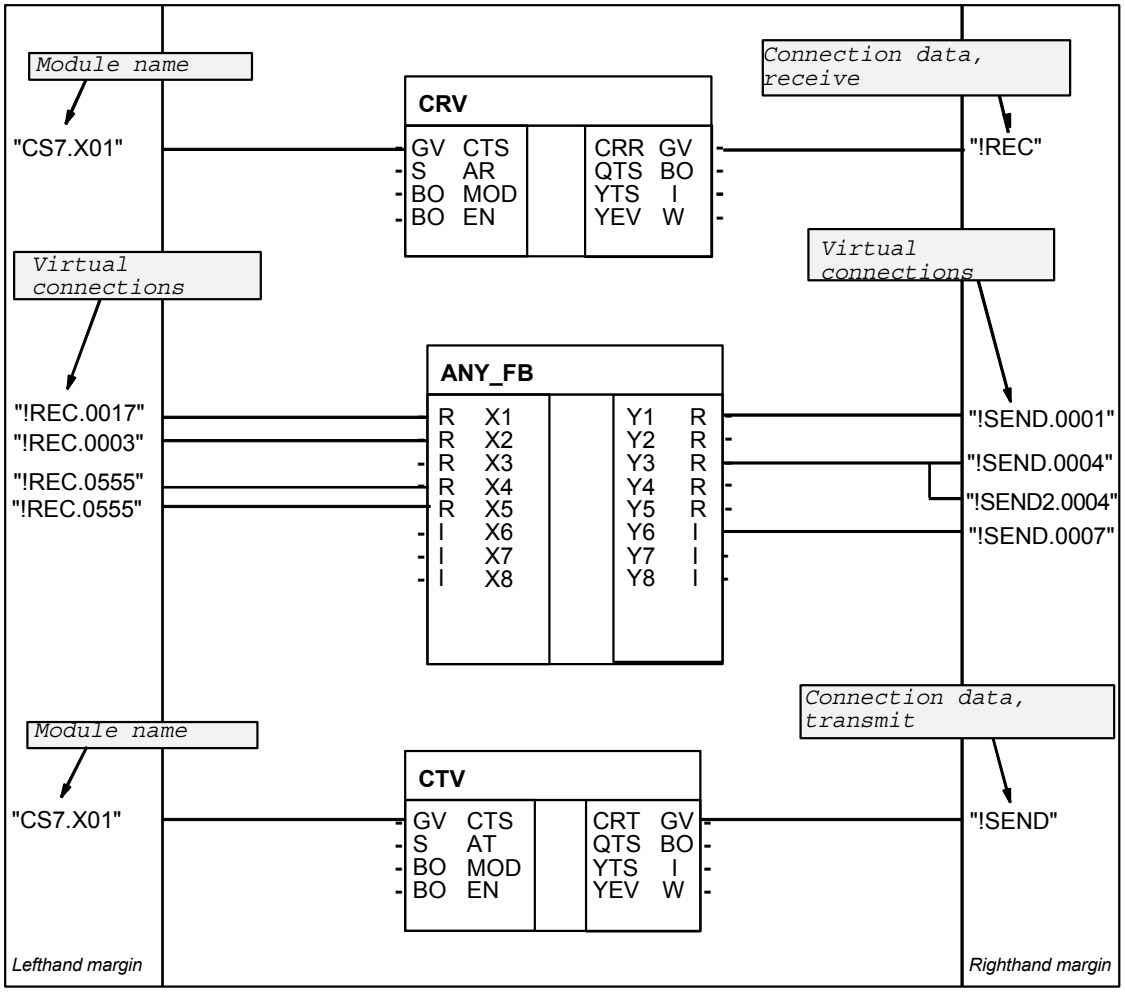


Fig. 3-83 Configuring: Receiving and transmitting with virtual connections

Configuring rules with reference to the example:

- The virtual connections, which belong to a virtual connection name, can be configured at block I/O with different data types (ANY_FB.Y1 "REAL" and ANY_FB.Y6 "INTEGER") and in any sequence.
- Virtual connections (receive) at block inputs can be configured a multiple number of times if the inputs have the same data type. These inputs are supplied with identical data. (ANY_FB.X4 and X5)
- The same virtual connection (transmit) with identical connection name and sequence number may not be configured a multiple number of times at block outputs.
- Several different virtual connections (transmit) can be configured at a block output (ANY_FB.Y3). The connections can differ, both in the connection name as well as in the sequence number.

Telegram structure of the connection name/data "!REC" from the example:

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

Table 3-40 Telegram structure of the connection name/data "!REC"

Telegram structure of the connection name/data "!SEND" from the example:

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

Table 3-41 Telegram structure of the connection name/data "!SEND"

The structure of the configured telegram appears in similar form in the CFC reference data in the view "cross-references, operands" or in the CPU MAP listing (of the CFC) under "virtual connections". The configuring can be checked using these lists.

NOTE

- The virtual connection names are known on the CPU. Data from various function charts can be combined to form a telegram; however, this is not possible from various CPUs.
- Data is processed by the receive/transmit blocks in their sampling time. The sampling times of the blocks with virtual connections have no influence on the telegram processing cycle.
- The configuring engineer is responsible in ensuring that the telegram structure and length are compatible with that of the coupling partner (refer to the chapter Mode of operation of the couplings). These regulations are dependent on the secondary coupling. If an error situation develops, the receive/transmit block disables itself and makes an entry in the communications error field (e.g. PROFIBUS DP or subrack coupling), or, communications are not established (e.g. Industrial Ethernet).

3.12.1.2 I/O of the CRV, CTV blocks

Inputs CTS The configured coupling module name via which communications is to be realized, is specified at input CTS of the block. For CP50M0/CP51M1 modules, it is also necessary to specify the connector (X01 for CP50M0/CP51M1 or X02 for CP50M0).

Input AR, AT The address parameter for communications is specified at input AR, AT. It consists of a channel name and the optional address stages. The significance of the address parameters is dependent on the coupling used. (e.g. PROFIBUS or DUST).

Input MOD The data transfer mode is configured at the MOD input (e.g. "R" for refresh or "H" for handshake)

Input EN Input EN defines whether data is to be transferred in the current operating cycle.

Inputs CRR, CRT The virtual connection name, receive or send is configured at input CRR or CRT.

3.12.2 Channel marshalling blocks CCC4 and CDC4

Application Channel marshalling blocks are used to split-up or combine channels.

3.12.2.1 Group block CCC4

General The CCC4 function block (Communication Collect Channel 4) combines up to 4 channels to form one. The channels may have different address data, be located at different interfaces and have different data transfer modi as well as channel lengths.

Prerequisites In order that the function block can operate, at least 2 channels must be combined (CT1- and CT2 input data are mandatory).

Data entries at connections CT3, CT4 If only 2 channels are to be combined, then a "0" (zero) should be configured at initialization inputs CT3 and CT4. In this case, connections AR3, AR4, MO3, MO4, LT3 and LT4 are no longer evaluated.

Data entries at input CTS, AT, MOD The transmit channel is specified at inputs CTS, AT and MOD. The length of the net data to be transmitted is obtained from the sum of the receive data. Receive channels 1-4 are combined, one after the other to form a large net data block.

Example

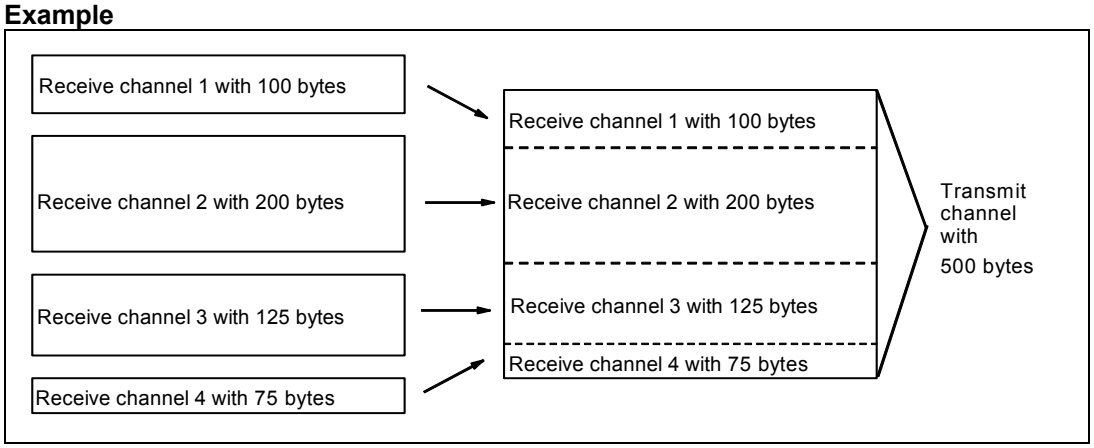


Fig. 3-84 Combining 4 receive channels to form a transmit channel

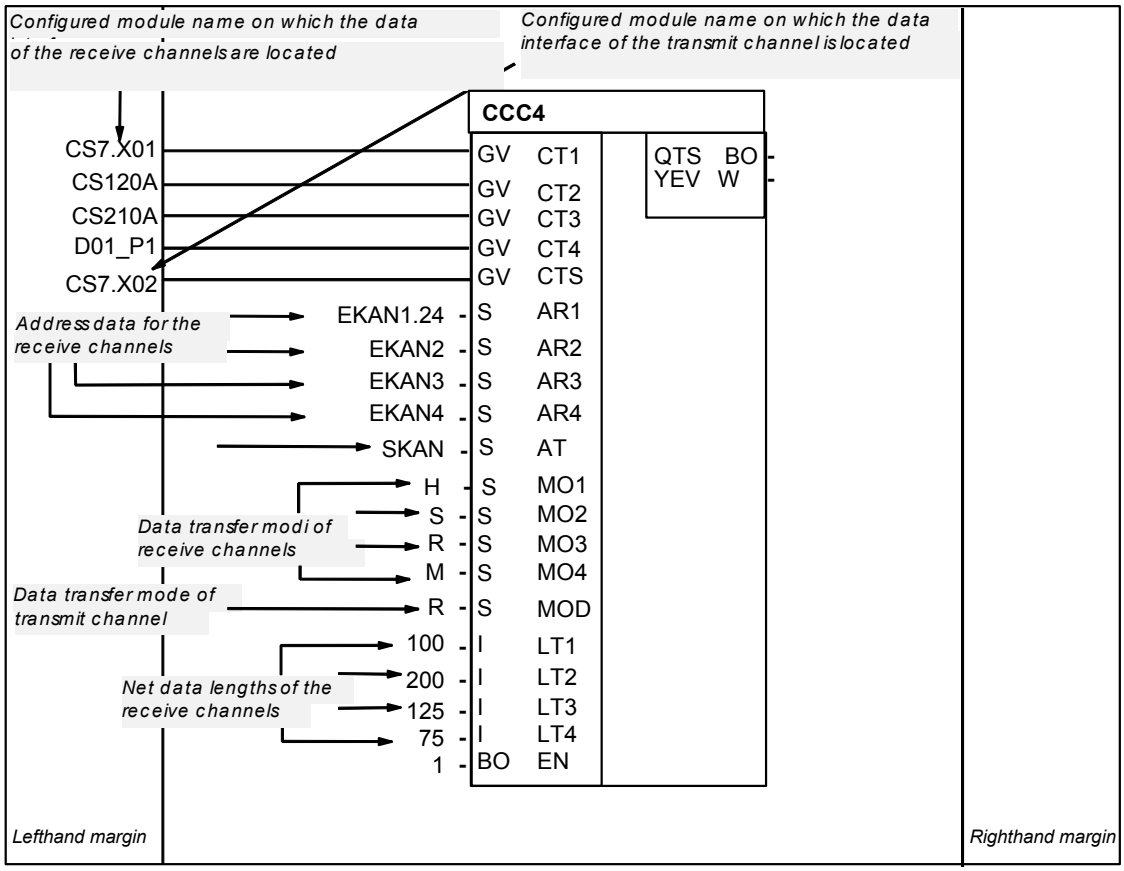


Fig. 3-85 Configuring example: CCC4 connections when combining 4 channels

3.12.2.2 Distribution block CDC4

General

The function block CDC4 (Communication Distribute Channel 4), subdivides a channel in up to 4 channels. The channels may have different address data, be located on different data interfaces, and have different data transfer modi as well as channel lengths.

Prerequisites

In order that the function block can operate, the receive channel must be sub-divided into at least 2 transmit channels (CT1- and CT2 input data are mandatory).

Data entries at inputs CT3, CT4

When sub-dividing a channel into only 2 channels, a "0" (zero) must be configured at initialization inputs CT3 and CT4. AR3, AR4, MO3, MO4, LT3 and LT4 are, in this case, no longer evaluated.

Data entries at inputs CTS, AT, MOD

The receive channel is specified at inputs CTS, AT and MOD. The length of the net data to be received is obtained from the sum of the transmit data.

NOTE

If one of the transmit channels is configured in the handshake mode and precisely this channel is not read-out on the receive side, then the CDC4 function block cannot operate until this one channel has been read-out. In this case, the block is temporarily inhibited.

3.12.2.3 Compatible net data structure

For blocks CCC4 and CDC4, the net data are unstructured (data type, octet string). Thus, they are compatible to any net data structure. In order that the transmitter and the associated receiver can correctly synchronize with one another, only the net data lengths must be identical.

3.12.3 Diagnostic outputs

General

After each processing cycle, the result of the processed data interface(s) is output at the YEV output of the transmit- and receive blocks as well as the channel marshalling blocks (CTV, CRV, CCC4, CDC4). The YEV output is the WORD type; the 16 bits are sub-divided into three areas:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Channel statuses (only CCC4, CDC4)				Channel assignment (only CCC4, CDC4)				Fault cause							

Table 3-42 Diagnostic outputs

3.12.3.1 Fault/error cause

Hexadecimal value

The possible fault/error cause is displayed in bits 0-7 in the form of a hexadecimal value (this should not be evaluated bit-coded):

Hex. Value	Significance	Counter-measure
0	No fault/error, data transfer was successful.	
1	Block was permanently disabled after initialization due to a configuring error or after an internal error (refer to the communications error field or YTS output for detailed information).	Correct configuring.
2	Communications partner not ready or the communications link was physically interrupted (refer to YTS for more detailed information).	Check coupling partner, cables and connector.
3	Communications partner is not transmitting/receiving (depending on the enable input of the communications partner). The function block is not transmitting/receiving because the communications partner has signaled that it is not transmitting data.	Activate the communications partner
4	Only for transmitters: Data cannot be transmitted. Normally, in the handshake/select mode: The communications partner has still not read-out the last data; seldom in the refresh mode: Communications partner is presently reading).	Configure the transmitter to be slower or the receiver to be faster.
5	Only for receivers: No new data could be received (the communications partner hasn't transmitted any new data since the last data was received).	Configure the receiver to be slower or the transmitter faster.
6	Inconsistent data (subrack coupling: when shutting down the master subrack)	None (proceed with new initialization)
7	Only select transmitters: channel occupied. Another function block is presently transmitting.	All select transmitters can coordinate via the enable input.
8	Only multiple receivers: Reception erroneous. Data read-out took too long; in the meantime, the transmitter has already written new data into the channel.	Configure receivers in a faster (higher-priority) sampling time.
9	Still being initiated. Transmit/receive operation was therefore not able to be started.	

Table 3-43 Fault/error cause

Comments to numbers 4 and 5

In the handshake mode, these numbers can sporadically occur which is quite acceptable. This is because full synchronization between the communications partner is not always possible. Receivers and transmitters should operate approximately in the same cycle.

In the refresh mode, these numbers should not occur if the transmitter is always faster than the receiver.

3.12.3.2 Channel assignment

General information

The area is only used by the CCC4 and CDC4 function blocks. In this case, a number indicates which channel is involved with the error (bits 0-7). As the channel marshalling block can process up to five channels, the numbering is as follows:

Number	Channel
0	Main transmitter/receiver (corresponding to CTS-, AT- or AR-connection data).
1	Transmit/receive part 1 corresponding to the CT1- and AT1- or AR1 connection data)
2	Transmit/receive part 2
3	Transmit/receive part 3
4	Transmit/receive part 4

Table 3-44 Channel assignment

3.12.3.3 Channel statuses

General

The area is only used by function blocks CCC4 and CDC4. This indicates which channels are not operating error-free.

In the "channel statuses" range it is specified on which channel faults were identified when processing the channel

This area is bit-structured:

- 1=no fault
- 0=fault

Bit	Channel
11	Transmit/receive part 1
12	Transmit/receive part 2
13	Transmit/receive part 3
14	Transmit/receive part 4
15	Main transmitters/receivers

Table 3-45 Channel statuses

3.12.4 Introduction – "Pointer-based communication blocks"

Up to D7-SYS Version 6, serial or parallel data transfer operations for SIMATIC control systems were configured using the so-called "virtual communication couplings" methods (shown in CFC charts e.g.: "!VNAME.0001").

Exception: The fiber-optic cable drive coupling SIMOLINK is configured using special SIMOLINK blocks.

From D7-SYS Version 6, communication links, for example PROFIBUS-DP, SIMATIC-CPU ↔ FM 458-1 DP as well as for SIMATIC TDC or T400 and SIMADYN D can be alternatively configured using communication blocks which have become recently available.

In this case, interface data is accessed from the CFC screen using new blocks, which are inter-connected using a special pointer interface.

Both of these configuring methods (virtual interconnections and pointer-based communications) can be used together on the same hardware platform, in the same configuring (application software) and even for the same interface.

3.12.4.1 Principle mode of operation

Telegram blocks (CRV_T, CTV_P and S7RD_P, S7WR_P) allow access to the receiving or to the sending data blocks (telegrams) by providing a pointer to the particular data block.

This pointer is connected to read/write blocks (DRD…, DWR…). Together with an offset, a write block can save the data at its input connection at the required location in the buffer. A read block then retrieves the appropriate data from the specified location of the receive buffer and makes it available at its output.

This means that in principle, a virtual interconnection is replaced by a (read/write) block and a "normal" CFC connection.

3.12.4.2 Applications

Large data quantities

Pointer-based communications are especially advantageous where large amounts of data are involved. For large amounts of data, it is simpler and faster to configure and change and interconnections are more flexible.

Access to the I/O area (P bus) for FM 458-1 DP

128 bytes can be transferred from the FM 458-1 DP to the S7-CPU in each direction via the I/O area of the P bus.

Using the new S7RD_P/S7WR_P blocks, all 128 bytes can be copied into a buffer using a block and that with an optimized computation time. This buffer can then be accessed flexibly using read/write blocks via the pointer interface. Indexed access is also possible

Sub-areas can also be accessed using offset and length data.

Saving data in a data block

Data can be saved in a data memory which can be universally used. This data memory can then be accessed using read/write blocks via a pointer interface. Several similar buffers can be set-up in this data block. This means, for example, that recipes can be easily saved and called-up.

3.12.4.3 Features of pointer-based communications

When generating CFC charts, the configuring time and costs are reduced,
especially if very many virtual connections had to be generated.

Connections to the telegram data can be newly inserted and changed online (pointer, buffer offset).

Communication connections can be copied with or within chart blocks and centrally changed with them. This means that it is especially simple and quickly to configure, for example, similar communication links to a large number of drives.

Telegram buffer data can be accessed indexed using 2 offset data. This means that extremely simple modular programs (e.g. chart blocks) can be generated and used.

Larger data quantities can be transparently processed (e.g. blockwise) (copied), e.g. using the copy block CPY_P in data block DB_P.

For FM 458-1 DP:

using "B-Receive" (BRCV) high quantities of data can be transferred from the S7-CPU to the FM 458-1 DP via the K bus.

128 bytes can be simply configured and quickly transferred with low computation overhead via the I/O area of the P bus.

A special read/write block is available for every data type (BYTE, INT, DINT, REAL).

Before accessing REAL data, the type is checked.

For all platforms and interfaces of the SIMATIC control systems

These configuring possibilities can be principally used for all of the SIMATIC control system platforms. This means FM 458-1 DP, SIMATIC TDC, T400 and SIMADYN D. The reason for this is that block processing is independent of the subordinate (secondary) hardware.

For the same reason, this type of block communications can be principally used for all types of serial and parallel data transfer routes, where today "virtual communications" are used.

3.12.4.4 Associated function blocks

The blocks which can be used are arranged under the family names "ZeigrKom" or "PointCom" in the CFC block Catalog.

In order to be able to simply identify and easily assign to this block group, the blocks, whose function already corresponds to existing blocks, and which now output a pointer for this application, have a "**_P**" (pointer) at the end of the name.

Type name	Function
CPY_P	Copying buffer areas
CRV_P	Telegram block, receive (interface processing)
CTV_P	Telegram block, send (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

Type name	Function
DRD_8D	Data Read 8*DINT
DRD_8I	Data Read 8*DINT
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
DWD_BY	Data Write BYTE
S7RD_P	Receive 128 bytes via a P bus (only for FM 458-1 DP)
S7WR_P	Send 128 bytes via a P bus (only for FM 458-1 DP)
BRCV	Block data receive via S7 connection (only for FM 458-1 DP)

3.12.4.5 Pointer interface

For pointer-based communications, **a pointer is transferred to the telegram data buffer** between the blocks involved:

This pointer is actually a pointer which includes a structure, which in addition to the pointer to the net data also has information for monitoring purposes. This data includes, for example, the sampling time, block class, byte/word swap. It has the connection comment "ZeigPuffer".

3.12.4.6 Configuring information and instructions

The telegram blocks as well as the read/write blocks must be configured in **the same sampling time** in order to ensure consistency (this is checked when initializing).

Offset data must be carefully entered.
a) For pointer-based communications, the configuring engineer must precisely observe the offset (in bytes) of the 16-bit value (INT) or 32-bit value (REAL, DINT) to be addressed.
b) The offset must always be smaller than the buffer size.
Before accessing buffer data, a check is made as to whether the area (range) has been exceeded because of an offset which has been set too high.

If data is transferred to a PROFIBUS-DP station or to a SIMATIC CPU, then bytes (for INT) and, where relevant, words of the value to be transferred (for REAL, DINT) must be swapped.
The read/write blocks have a "Swap" connection – SWP – for this specific purpose.

In order to transfer telegrams via an interface, initially, it is sufficient to just configure the telegram block with the appropriate lengths data (CRV_T, CTV_P and S7RD_P, S7WR_P). Read/write blocks still don't have to be configured. This means that the interface can be tested or the computation time load through the interface configured using, for example, few resources.

3.12.4.7 Examples of CFC screenshots

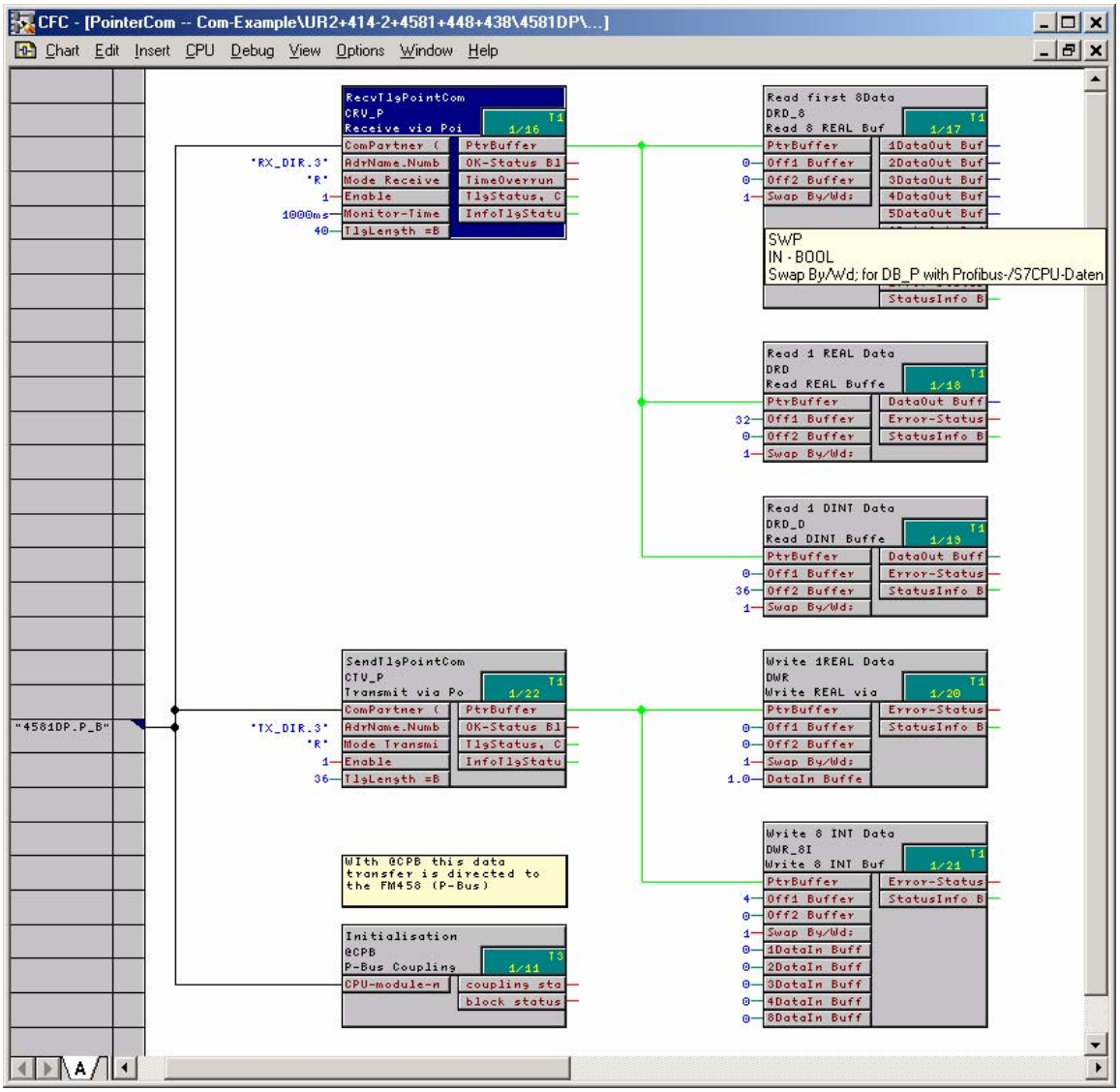


Fig. 3-86 CFC screenshot: Data transfer with telegram blocks and read/write blocks; here, for the interface P bus of the FM 458-1 DP (@CPB); bytes/words must be swapped due to the data management on the SIMATIC-CPU: SWP(Swap)=1

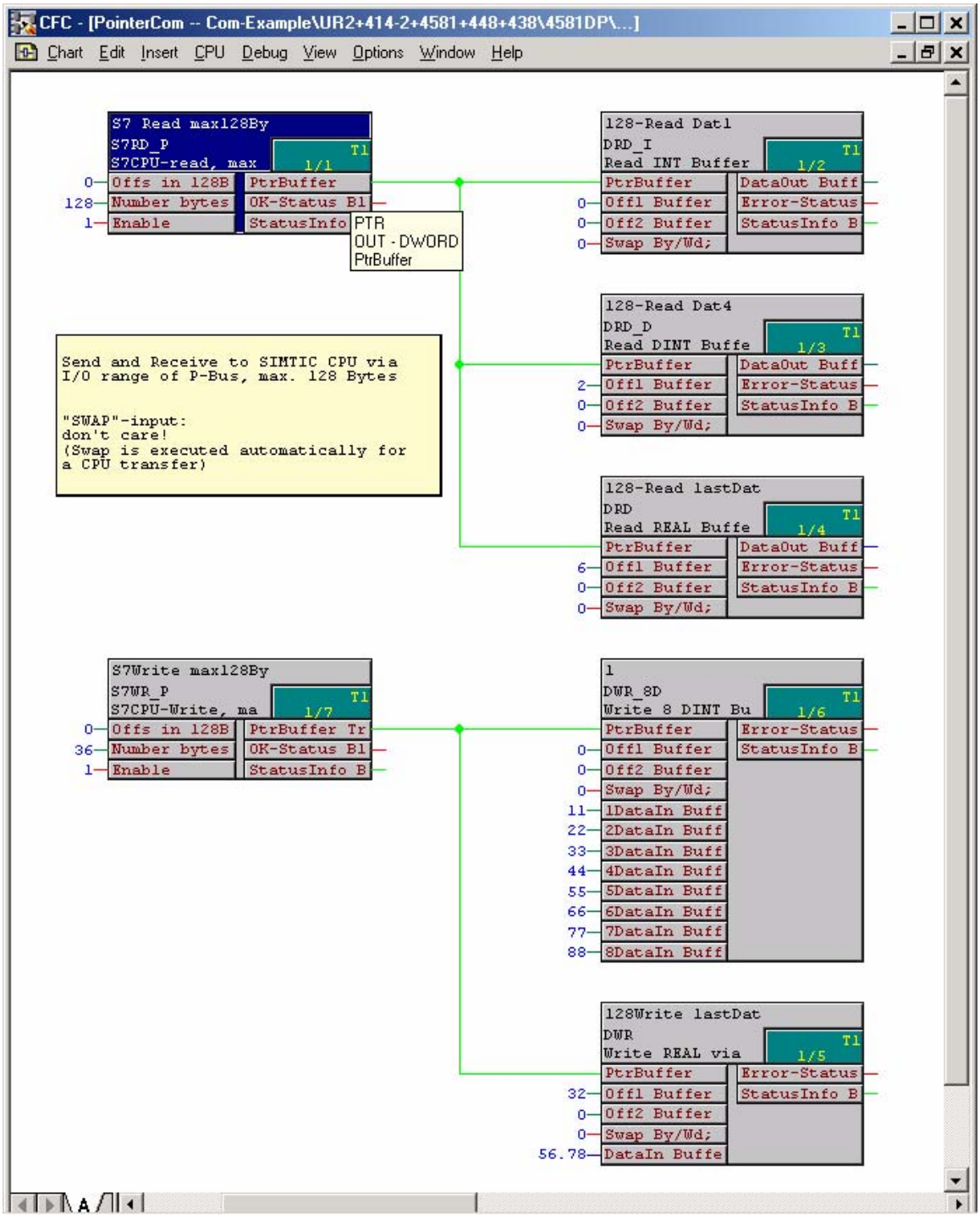


Fig. 3-87 CFC screenshot: Data transfer SIMATIC-CPU ↔ FM 458-1 DP via P bus I/O area

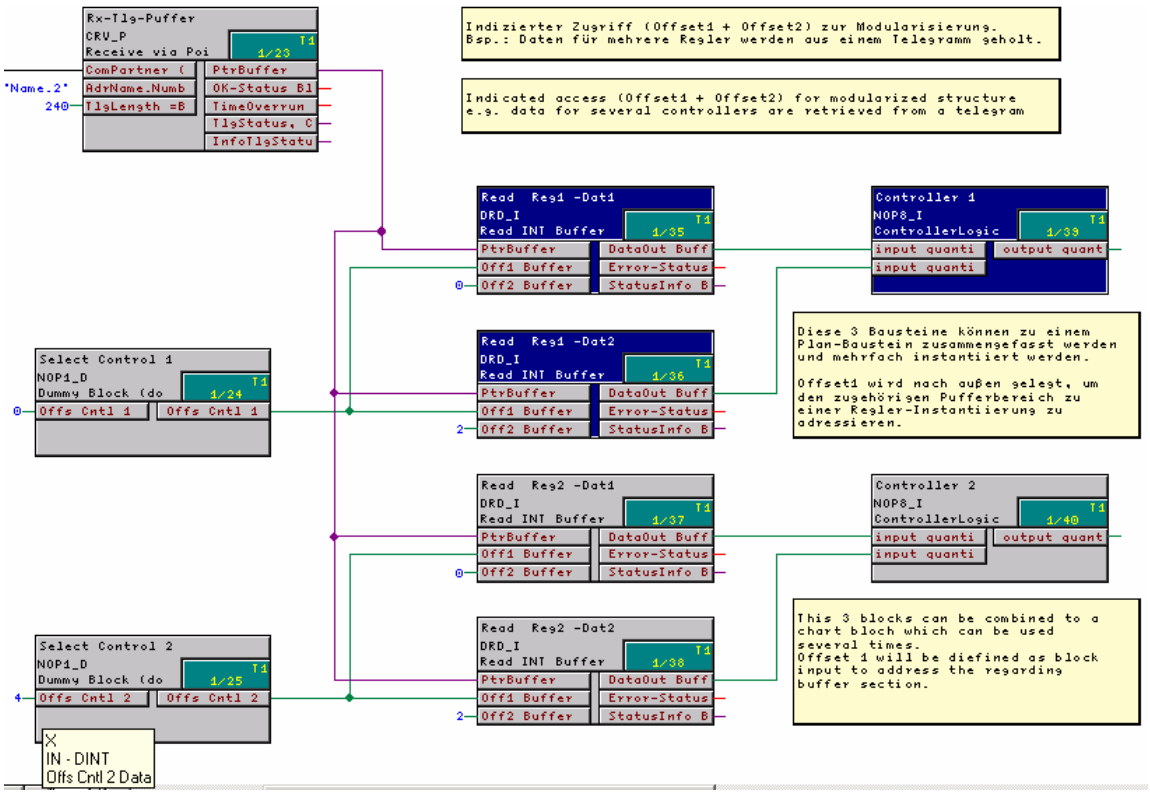


Fig. 3-88 CFC screenshot: Indexed addressing of the telegram data with 2 offsets

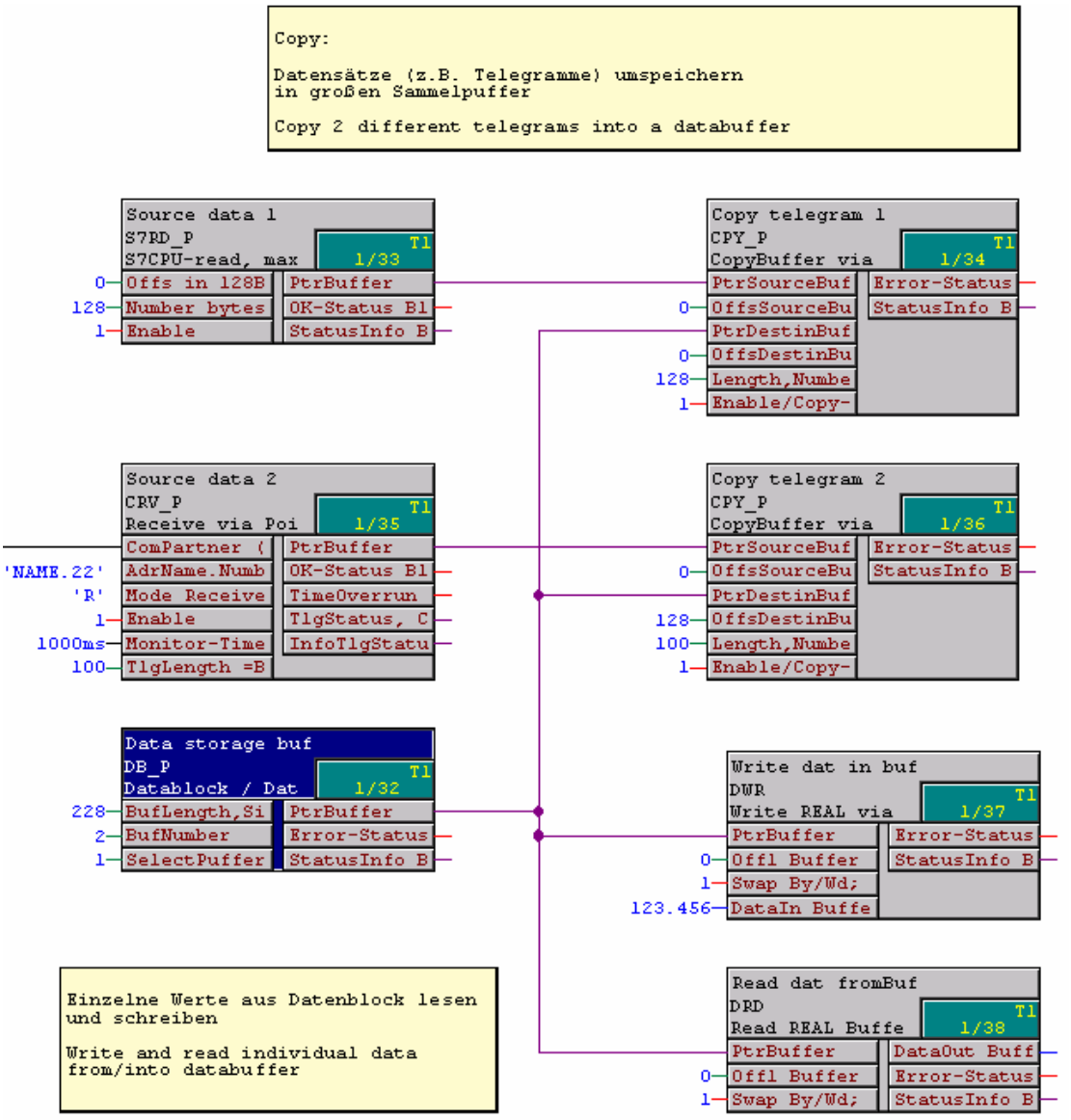


Fig. 3-89 CFC screenshot: Re-saving 2 received telegrams in a data block and single accesses to the data memory

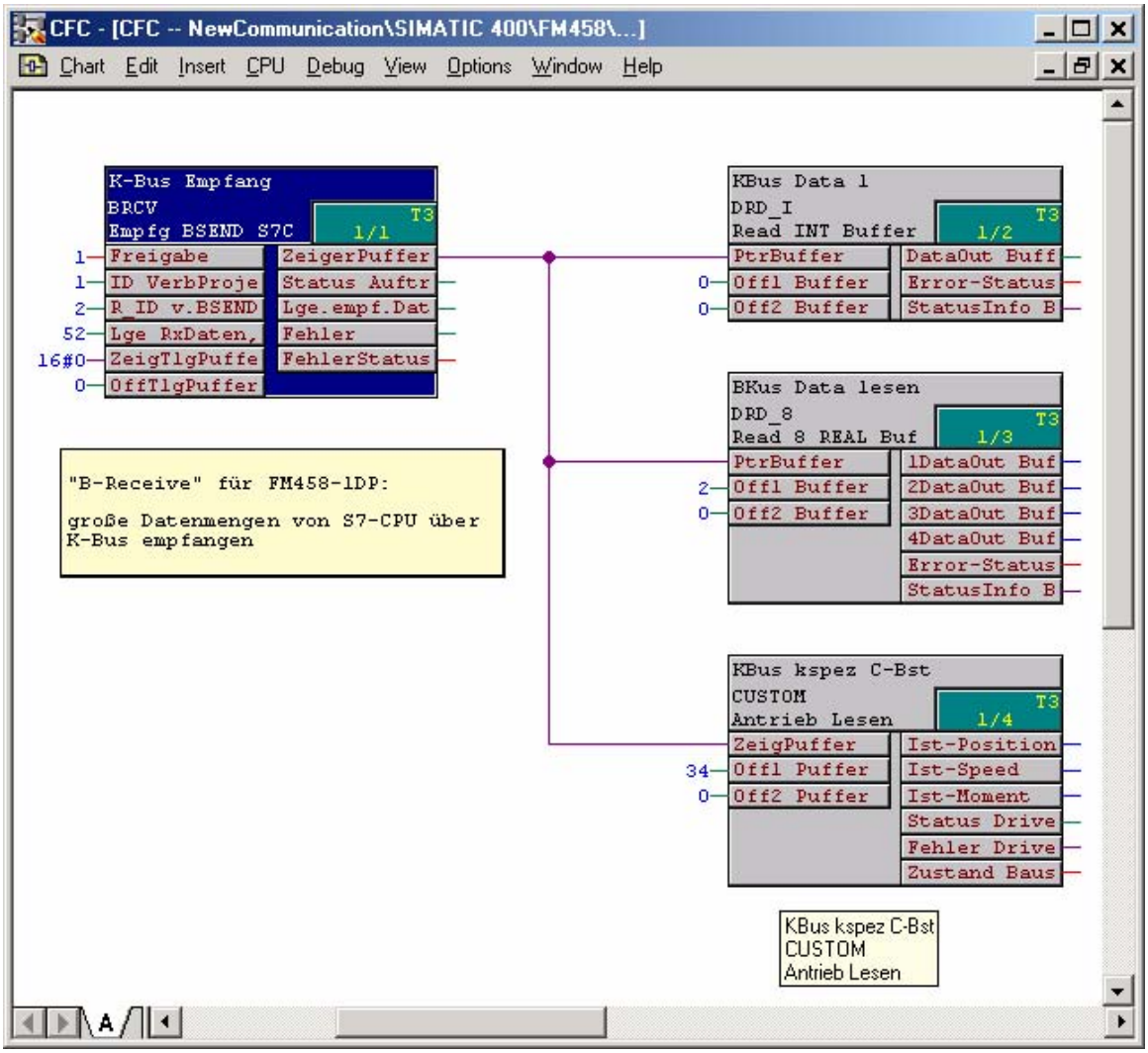


Fig. 3-83-90 CFC screenshot: Large data quantities received from a SIMATIC CPU via K bus using BRCV

3.13 Communications utility service

Brief description

- Provides a pool of information functions so that the user has access to system information on the CPU.
- Resource for start-up (commissioning) and debugging.

Start-up

The configured data (setpoints/actual values) are displayed and/or changed here, as well as the software optimized (interconnections, controller times modified.

Debugging

Causes of system faults (crash, run-up problems) and disturbances, where the cause is within the CPU module, can be determined here.

All of the communication utility, service activities are controlled via tasks, which are received via a coupling (corresponding to the data entries at the CTS and US inputs).

Operator control devices for the communications utility, service:

- Windows PC with CFC (e.g. in the test mode)
- Windows PC with SIMATIC Manager

Local service

Using CFC, SIMATIC Manager or the basic service tool, it is possible to access a CPU via the local RS232 interface of the CPU. No additional configuring is required.

NOTE

You can read-out the CPU module information using the CFC and the SIMATIC Manager.
Additional information
ont he CPU module, refer to the User Documentation "SIMATIC TDC, Basis software D7-SYS", Section "Diagnostics".

Central service

Each CPU of this subrack can be accessed via MPI or Industrial Ethernet coupling configured in the subrack.

The following must be configured:

- One per subrack:
  - MPI coupling: One CP50M1-/ CP50M0 module and a central block MPI coupling "@MPI"
  - Industrial Ethernet coupling: CP51M1 module and a central block „@TCPIP“
- At least one per CPU:
  - "SER" service function block.

Additional information
Refer to the Chapter "MPI coupling" for details on the MPI couplings or refer to the Chapter "Industrial Ethernet (TCP/IP) coupling" for details on the Industrial Ethernet couplings.

3.13.1 Function block SER

Data entries at the connections

"SER" function has a coupling connection. It can be configured several times for each CPU.

The **CTS** input designates the coupling module and the interface via which an operator control device is connected.

A channel name and address stage 1 is specified at input **US**.

- **Channel name**
  - max. 6 characters
  - ASCII characters with the exception of "point" and @
  - the channel name on a data interface must be unique.
- Enter "." after the channel name
- **Address stage 1**
  - CPU slot number. The operator control program addresses the CPU via this number.
  - The data entry must have two digits: e.g. "01", "02", ..., "24".

Example: Configuring with CFC

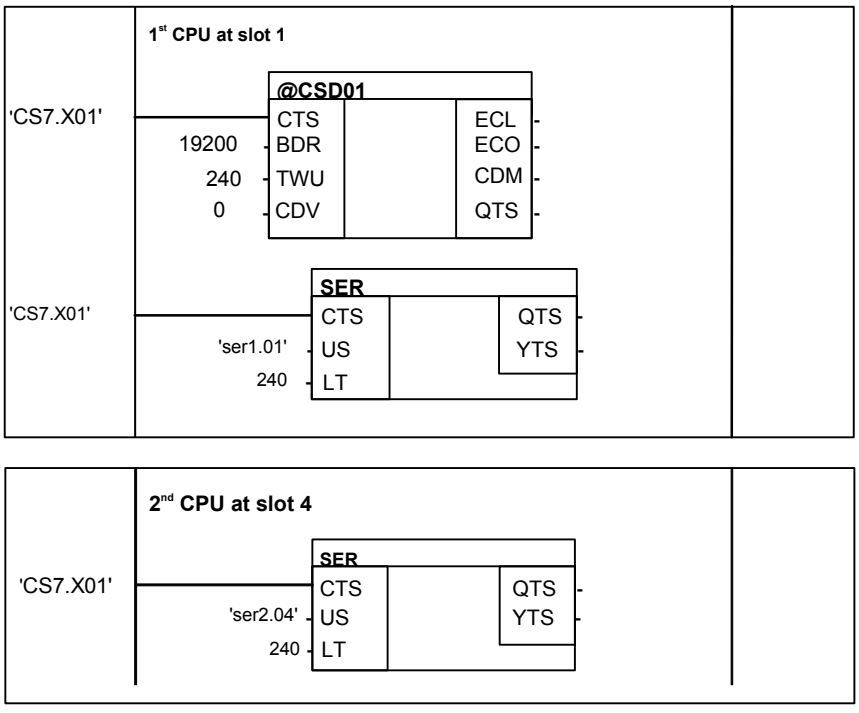


Fig. 3-91 Example: Configuring with CFC

3.13.2 System load, response times

General

Service is actually processed in a sampling time of approximately 32 ms. (The sampling time, specified at the SER blocks is therefore not decisive for processing.) In the sampling time used, the service blocks have a certain computation time available, and more precisely, a maximum of one basic clock cycle (T0).

NOTE

The ratio of the basic clock cycle T0 to the sampling time used defines the CPU performance available and therefore the system load.

Example 1

Basic clock cycle T0=1ms; selected sampling time=32ms

- Every 32 ms, 1 ms is reserved for the service utility
- System load=1 ms / 32 ms=0.03125=3.125 %

Example 2

Basic clock cycle T0=2ms; selected sampling time=16ms

- Every 16 ms, 2ms is reserved for the service utility
- System load=2 ms/16 ms=0.125=12.5 %

Computation time

The computation time available is evenly distributed among all of the service blocks (there is no priority). This means, that as long as time is available, if possible, all SER blocks are executed once. An SER block processes a maximum of one task per each clock cycle. If the reserved computation time isn't fully used, for example, as there is no task to process, then this computation time is made accessible to the system.

Resource distribution

For multiple configuring 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").

3.14 Communications utility time of day synchronization

General

The communications utility, time of day synchronization allows a unified system time to be provided over several SIMATIC TDC subracks.

Time

The following can be used as time source:

- the CPU inserted to the far left in a subrack
- the CP52A0 / CP53M0 communications buffer module
- The industrial Ethernet module CP51M1

The time is distributed:

- within a SIMATIC TDC subrack via a communications buffer module
- to other SIMATIC TDC subracks via the subrack coupling

Function block

Precisely one function block RTCM should be configured per subrack to distribute the system time.

Further information
to configure function blocks, refer to the user documentation "SIMATIC TDC, Function Block Library".

The following function blocks are used to read-out the system time:

- RTCABS: absolute time in the date/time of day format
- RTCREL: relative time in seconds since 01.01.88

These blocks can be configured as required.

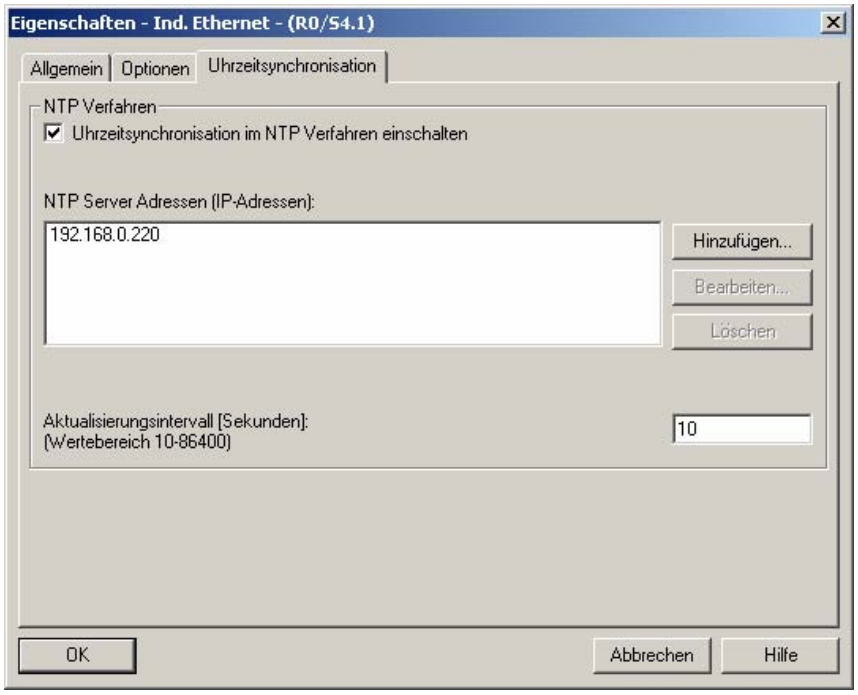
CP51M1

The CP51M1 IE module is in a position to actively retrieve the time from up to four NTP time servers (e.g. SICLOCK TM).

Additional information
on the NTP clock synchronization technique, refer to: www.ntp.org

The following settings are required in HW config under the tab "Time synchronization" of the "Ind. Ethernet connection of the CP51M1“:

- Enable the NTP technique
- Add the IP address of the NTP server or addresses of the servers (if there are several)
- Set the update interval (seconds)



The CP51M1 module retrieves the actual time from all configured NTP servers in the update interval (10 seconds is recommended). The time of the actual (best) NTP server is used. This means that the time supplied from all of the configured NTP servers cannot noticeably differ (<< 1 ms). If the time of the NTP servers deviate between one another, then “jumps” are possible in the time. The "best" NTP server is continually determined using among other things the runtime of the time interrogation.

Note: If more than one NTP server is configured, then only the NTP servers are used for the time synchronization that calibrated themselves with a reference clock (e.g. DCF77 or GPS) within the last 24 hours. If only one NTP server is configured, then this is used even if it was not synchronized with a reference clock.

Entries for important events are made in the CP51M1 diagnostics buffer – for example:

- Synchronization with an NTP server (coming/going)
- NTP server is not accepted
- Loss of the synchronization of the NTP server with reference clock

Note: If an SNTPR-FB has been configured for this CP51M1 and in addition an NTP, then the time, determined by SNTPR is distributed in the TDC rack. The time determined using the NTP technique is only used to set the module clock of the CP51M1 module. An appropriate entry is made in the diagnostics buffer.

3.15 Communications with SIMATIC Operator Panels

Introduction

A configuring engineer will be shown how to implement a coupling from SIMATIC TDC to a SIMATIC OP7 using this configuring software example.

NOTE

Proceed in a similar fashion when configuring couplings to the OP27, OP37 SIMATIC Operator Panels and the TP37 SIMATIC Touch Panel.

The example described here, includes all of the available SIMATIC TDC function blocks, and shows how they are essentially used. The functional scope of the configuring software example has been consciously kept extremely low, so that you can quickly get to grips with the subject. It is simply possible to expand the functionality and/or the hardware components. However, the information provided in the applicable function block documentation must be observed.

The designations used for data blocks, flags, variables etc. have been randomly selected, and are only binding for this particular configuration software example.

NOTE

- When saving values which have been changed using SIMATIC Ops, this is realized on the SIMATIC TDC CPU in the SAVE area.
- When the battery back-up fails, the configured value at the input is used as default.

Prerequisites

The structure of these configuring instructions represents the sequence of the various operating steps, with which the complete configuring software can be generated. However, it should only be considered as a recommendation, and is not mandatory.

We are assuming that you know how to handle the SIMATIC Manager (including HWConfig and CFC), configuring SIMATIC TDC as well as configuring OP7 with ProTool/Lite.

Literature which is available on these subjects:

- SIMATIC TDC User Manuals
- SIMATIC Equipment Manual OP7/17
- SIMATIC HMI, User Manual
  ProTool/Lite configuring software

3.15.1 Configuring example

Functional scope

The configuring software example supports the following OP7 functions:

- Reading and writing variables
- Output of operating messages
- Output of alarm messages including acknowledgment

- Interrogating the function keyboard
- Updating date and time

Hardware

The following equipment and components are selected and located as follows for the configuration example:

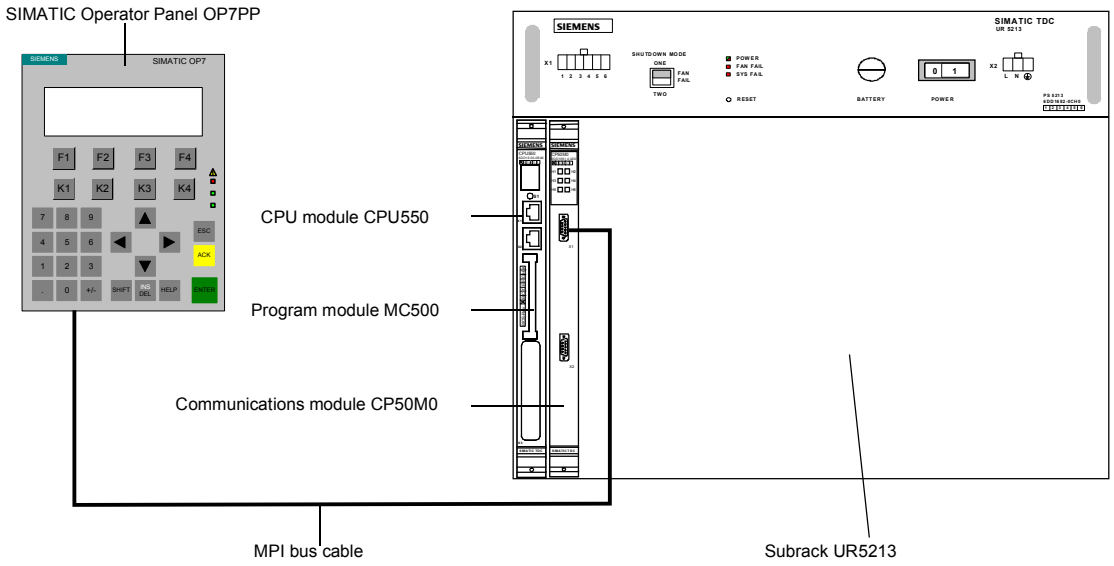


Fig. 3-92 Setting-up the configuring example

3.15.2 Configuring SIMATIC TDC

General information

All of the configuring, which involves SIMATIC TDC, is made in the SIMATIC Manager. The work is divided into the "Selecting components in HWConfig" and "Configuring with CFC" sections.

3.15.2.1 Selecting the components in HWConfig

The configuration example is configured in HWConfig. The standard program inputs can be accepted. The only changes involved:

- Sampling time **T4** of the CPU550 = **64ms**
- Highest MPI address of the MPI interface onto CP50M1 / CP50M0 = **126** (126 is entered as standard in ProTool/Lite)
- The first interface as MPI configured

3.15.2.2 Configuring with CFC

After executing "Save and compile" in the "HWConfig", the "D01_P1" symbol was inserted in the SIMATIC Manager below the "SIMADYN D station".

Inserting a new chart

A new chart, called "OP7" is added, in the associated chart container, to the existing charts "@SIMD1" and "@SIMD2". All of the additional configuring work will now be made in a new chart called "OP7".

Agreements

- All of the function blocks to be configured are configured in the sequence level T4.
- If not explicitly listed, the standard assignments of the function block connections are kept.
- Only the relevant connections are listed in the following configuring tables.

3.15.2.2.1 Initializing the OP7

Brief description

The function blocks **@MPI** and **S7OS** are connected to the configured coupling module (first interface onto CP50M1/ CP550M0) via the **CTS** inputs. This establishes the connection between SIMATIC TDC and OP7.

Configured software

FB	Connection	Connection assignment (significance)
@MPI	CTS	**D0200C.X01** (global operand, module name)
S7OS	CTS	**D0200C.X01** (global operand, module name)
	US	**testop.01** (address parameter)

Table 3-46 Connection assignment @MPI and S7OS

3.15.2.2.2 Reading function block connections (I/O)

Brief description

A counter was configured for this function, which continually increments from the initial value ("0") up to a final value ("50"). It then automatically resets itself and starts again from the beginning. Output **Y** (counter status) of the **CTR** is interlocked with a global operand (OP connection), whose contents are read-out at OP7.

NOTE The flag No., specified under SIMATIC TDC for the OP connection, must also be assigned the configured variables under ProTool/Lite.

Configuring software

FB	Connection	Connection assignment (significance)
BF	T	**500ms** (time constant)
CTR	LU	**50** (counter upper limit)
	Y	Symbol name: **Z_output** Flag No.: **MW10** (global operand, OP connection)

Table 3-47 Connection assignment, BF and CTR

3.15.2.2.3 Writing function block connections

Brief description

A value from OP7 is read-in using a global operand (OP connection), fed through a dummy block (NOP1_I), and is sent back to the OP7 with an additional global operand (OP connection); it is read-out from the OP7.

NOTE

The flag No. for the OP connections, specified under SIMATIC TDC, must also be assigned the configured variables under ProTool/Lite.

Configured software

FB	Connection	Connection assignment (significance)
NOP1_I	X	Symbol name: **OP_SOLL** Flag No.: **MW20** (global operand, OP connection)
	Y	Symbol name: **OP_IST** Flag No.: **MW30** (global operand, OP connection)

Table 3-48 Connection assignment, NOP1_I

3.15.2.2.4 Configuring events

Brief descriptions

If the counter starts a new count loop, an event is output. Output **QO** of function block **CTR** outputs the signal. This signal is extended (FB PDF), converted from the "boolean" format into the "word" format (FB B_W), and transferred to function block **S7EMA** as the first event message word.

The S7EMA is assigned a virtual data block number for the user data area "event messages" via a global operand (OP connection).

NOTE

The data block No., specified under SIMATIC TDC for the OP connection, must also be assigned the configured area pointer for event messages under ProTool/Lite.

Configured software

FB	Connection	Connection assignment (significance)
PDF	I	Function block **CTR**, output **QO** (event message signal)
	T	**5000ms** (time constant)
B_W		(Conversion from boolean to word)
S7EMA	XDB	Symbol name: **BM** Data block No: **DB1** (global operand, OP connection)

Table 3-49 Connection assignment, @MPI and S7OS

3.15.2.2.5 Configuring alarm messages

Brief description

If the counter starts a new count loop a alarm message is output (at the same time as the event message) . Output **QO** of function block **CTR** supplies the signal. This signal is converted from the "boolean" format into the "word" format (FB B_W), and is transferred to function block **S7AMA** as the first event message word.

S7AMA is assigned a virtual data block No. for the user data area "alarm messages" via a global operand (OP connection).

NOTE The data block No. for the OP connection, assigned under SIMATIC TDC, must also be assigned the configured area pointer for alarm messages under ProTool/Lite.

Configured software

FB	Connection	Connection assignment (significance)
B_W	I1	Function block **CTR**, output **QO** (signal for the alarm message)
S7AMA	XDB	Symbol name: **SM** Data block No.: **DB10** (global operand, OP connection)

Table 3-50 Connection assignment, B_W and S7AMA

3.15.2.2.6 Configuring the function keyboard

Brief description

The configuring of the function keyboard includes, on the SIMADYN D side, only the **S7FKA** function block. The actual assignment of the key functions is realized under ProTool/Lite.

S7FKA is assigned, via a global operand (OP connection) a virtual data block No. for the user data area "function keyboard image".

NOTE The data block No., assigned under SIMATIC TDC, for the OP connection, must also be assigned the configured area pointer for the function keyboard under ProTool/Lite.

Configured software

FB	Connection	Connection assignment (significance)
S7FKA	XDB	Symbol name: **FK_Tast** Data block No.: **DB20** (global operand, OP connection)

Table 3-51 Connection assignment, S7FKA

3.15.2.2.7 Configuring the interface area

Brief description

The time and date of the OP7 is cyclically updated by SIMATIC TDC using this function. S7IA is assigned, via a global operand (OP connection) a virtual data block No. for the user data area "interface area" .

NOTE

The data block No., assigned under SIMATIC TDC, for the OP connection, must also be assigned the configured area pointer for the interface area under ProTool/Lite.

Configured software

FB	Connection	Connection assignment (significance)
S7IA	XDB	Symbol name: **SB** Data block No: **DB30** (global operand, OP connection)

3.15.2.3 Importing the symbol table

General information

While configuring the CPU in HWConfig an empty symbol table is automatically set-up, which will later accept the symbol names configured using CFC. The file with the symbol names must then be imported into the symbol table when the CFC has been configured.

Symbol editor

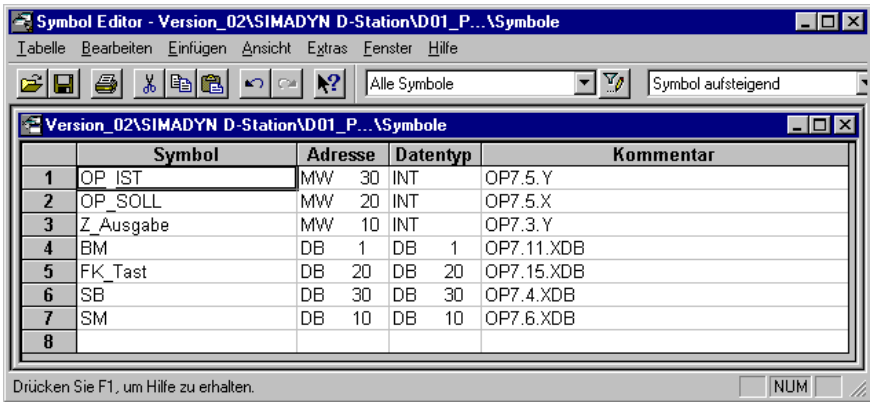
The symbol editor is opened from the chart container by double-clicking on "Symbols".

The symbol file (symbol.asc) is loaded in the symbol table using the menu command "Import table...".

NOTE

If changes are made in the symbol file in the CFC between two compilations, then a message to this effect is output. This message can also be taken from the actual memory path of the symbol file.

The following diagram shows the complete symbol table of the test software after having been imported:



	Symbol	Adresse	Datentyp	Kommentar
1	OP_IST	MW 30	INT	OP7.5.Y
2	OP_SOLL	MW 20	INT	OP7.5.X
3	Z_Ausgabe	MW 10	INT	OP7.3.Y
4	BM	DB 1	DB 1	OP7.11.XDB
5	FK_Tast	DB 20	DB 20	OP7.15.XDB
6	SB	DB 30	DB 30	OP7.4.XDB
7	SM	DB 10	DB 10	OP7.6.XDB
8				

Fig. 3-93 Symbol table with imported symbol file

The symbol table is saved and the operation completed using "Save table".

3.15.3 Configuring the OP7 with ProTool/Lite

General information

The configuring of OP7 is not described in detail here. If not explicitly mentioned, when configuring, the standard settings can be taken from ProTool/Lite.

NOTE

For error-free communications, it is absolutely necessary, that the flag- and data block numbers, configured in CFC, are transferred for the individual functions, unchanged, into ProTool/Lite.

Symbol table

CFC generates a symbol table, in which all of the flags and data blocks used are saved. This symbol table must be imported for the configuring work for ProTool/Lite.

The symbol names, configured in CFC for the OP7 configuring, can now be used in ProTool/Lite.

Configuring software

Configured software with displays (including variables to read and write values), event- and alarm messages as well as configured function keys must be generated for the OP7.

The following table provides an overview of the required configuring components with the associated values, harmonized and adapted to the CFC configured software:

Configured software	Setting
Control	SIMATIC S7-300/400
MPI settings	Communications partner slot: 1
Variables to read the function block connections (I/O)	Symbol name: Z_Ausgabe (VAR_1: Format "INT", type "A" area "M", MW10)
Variables to write into the function block connections (I/O)	Symbol name: OP_SOLL (VAR_2: Format "INT", type "E" area "M", MW20) Symbol name: OP_IST (VAR_3: Format "INT", type "A" area "M", MW30)
Area pointer, event messages	Symbol name: BM (DB1, DBW0, length "8" words)
Area pointer, alarm messages	Symbol name: SM (DB10, DBW0, length "8" words)
Area pointer, acknowledge PLC	DB10, DBW16, length "8" words
Area pointer, acknowledge OP	DB10, DBW32, length "8" words
Area pointer, function keyboard	Symbol name: FK_Tast (DB20, DBW0, length "1" word)
Area pointer, interface area	Symbol name: SB (DB30, DBW0, length "16" words)

3.15.4 Application information

3.15.4.1 Computation times

General information

The computation times of the function blocks are dependent on the application.

The computation times of the function blocks for an OP7 are listed in the following table. Each additional configured OP7 correspondingly increases the computation time.

	S7OS	S7EMA	S7AMA	S7FKA	S7IA
One OP7	120	2	33	22	18
Each additional OP7	55	2	33	22	18

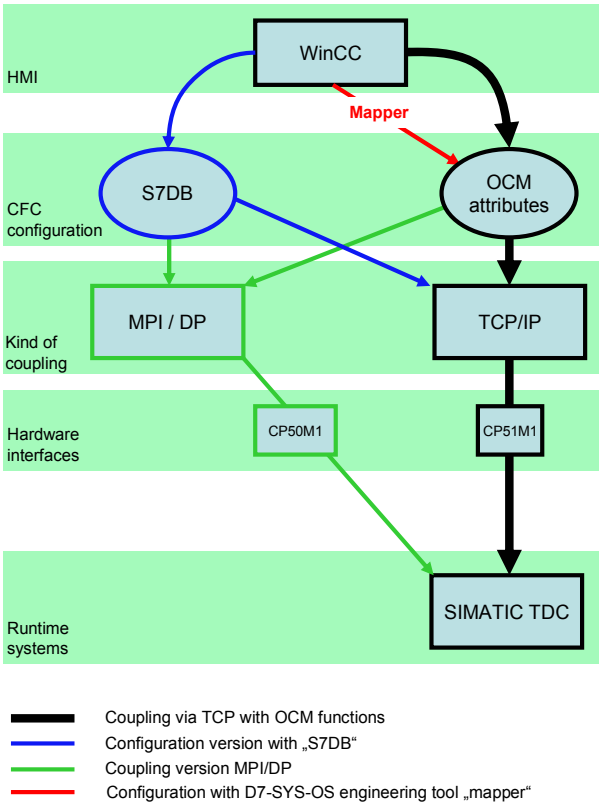
3.16 WinCC connection to SIMATIC TDC via the standard channel (SIMATIC S7 Protocol Suite.CHN)

The chapter describes the procedure when configuring the access to process variables (block connections) of a SIMATIC TDC CPU via the "standard channel" using WinCC. The various configuring and coupling options are shown in the following diagram.

The coupling via TCP/IP using the FB property “OCM” functions (Operator Control and Monitoring), when creating an address book, is completely described.
The special features of the configuration version with function block "S7DB_P" that supersedes to create an address book, is described in the following chapter.
An explanation is then given on how to use the MPI and PROFIBUS DP coupling types. (When using a PROFIBUS DP coupling for visualization purposes, it should however be taken into account that this coupling - generally used as fast drive coupling - is possibly slowed down by the HMI signals.)

The use of the D7-SYS-OS engineering tool (WinCC mapper) is then explained.

Overview – communication and configuring options
WinCC – SIMATIC TDC

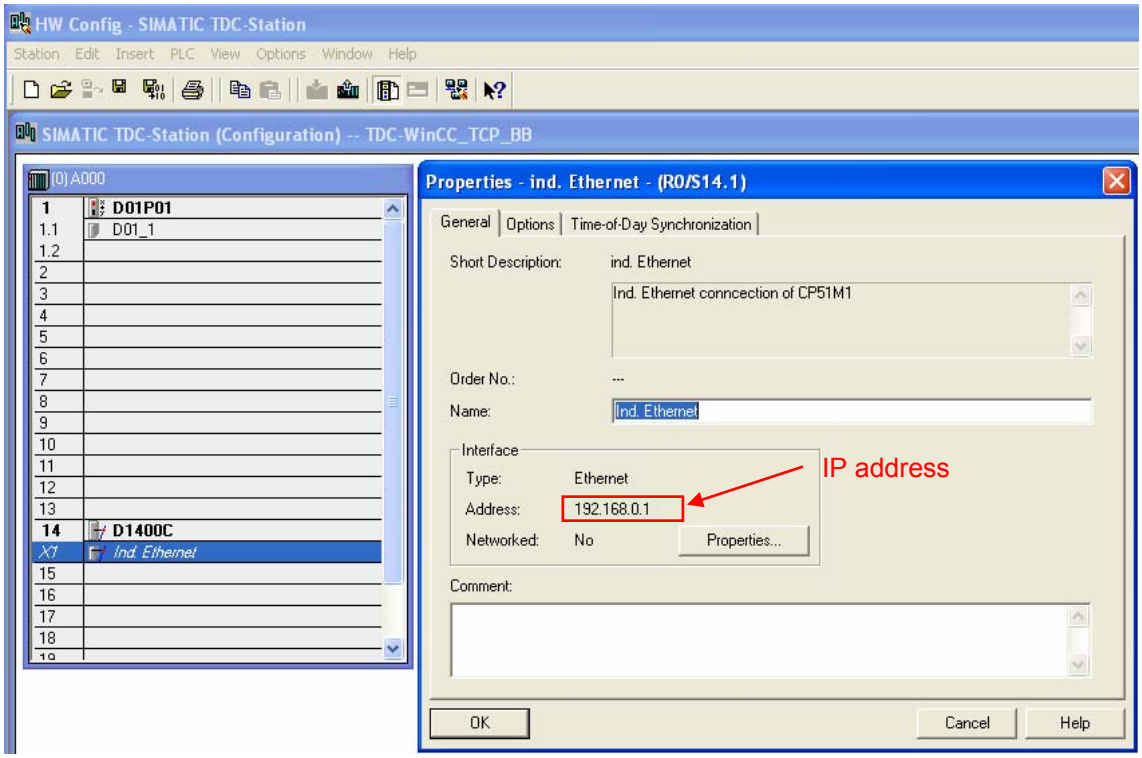


3.16.1 Coupling via TCP/IP with "OCM" functions

This chapter describes the procedure when configuring the access to process variables (block connections) of a SIMATIC TDC CPU using WinCC.

3.16.1.1 Configuring the coupling-relevant TDC hardware

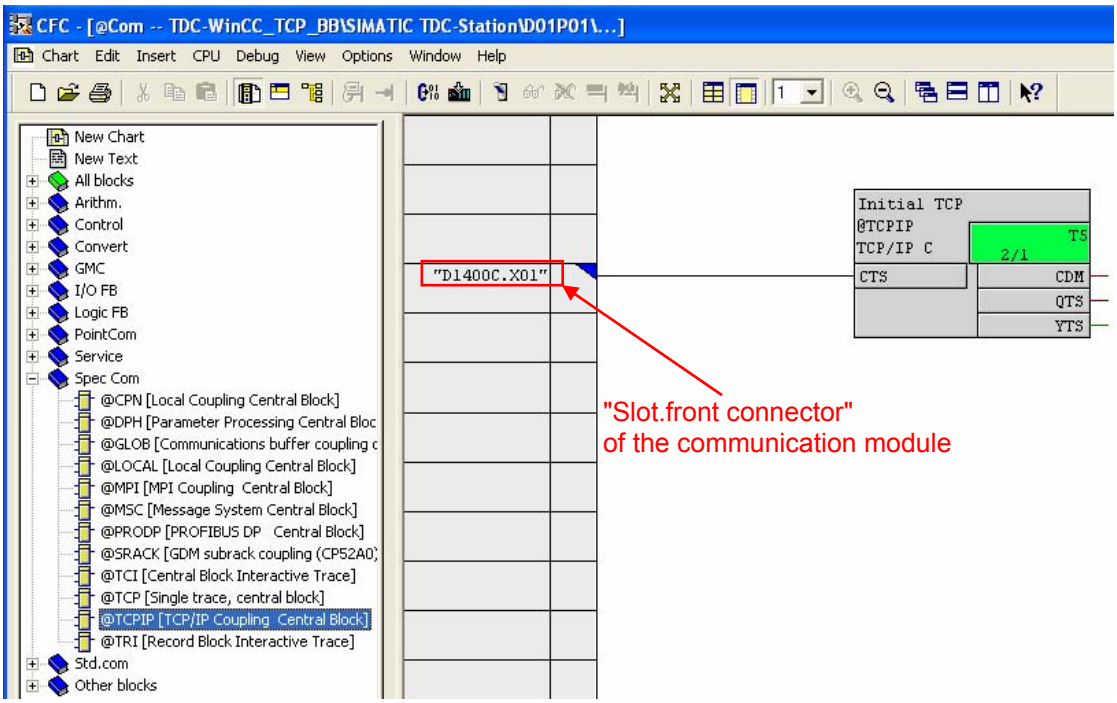
A CP51M1 should be configured as communication module in the SIMATIC TDC rack for the TCP/IP coupling.
Click on the properties button to open a window for inserting a subnet and defining the IP address.



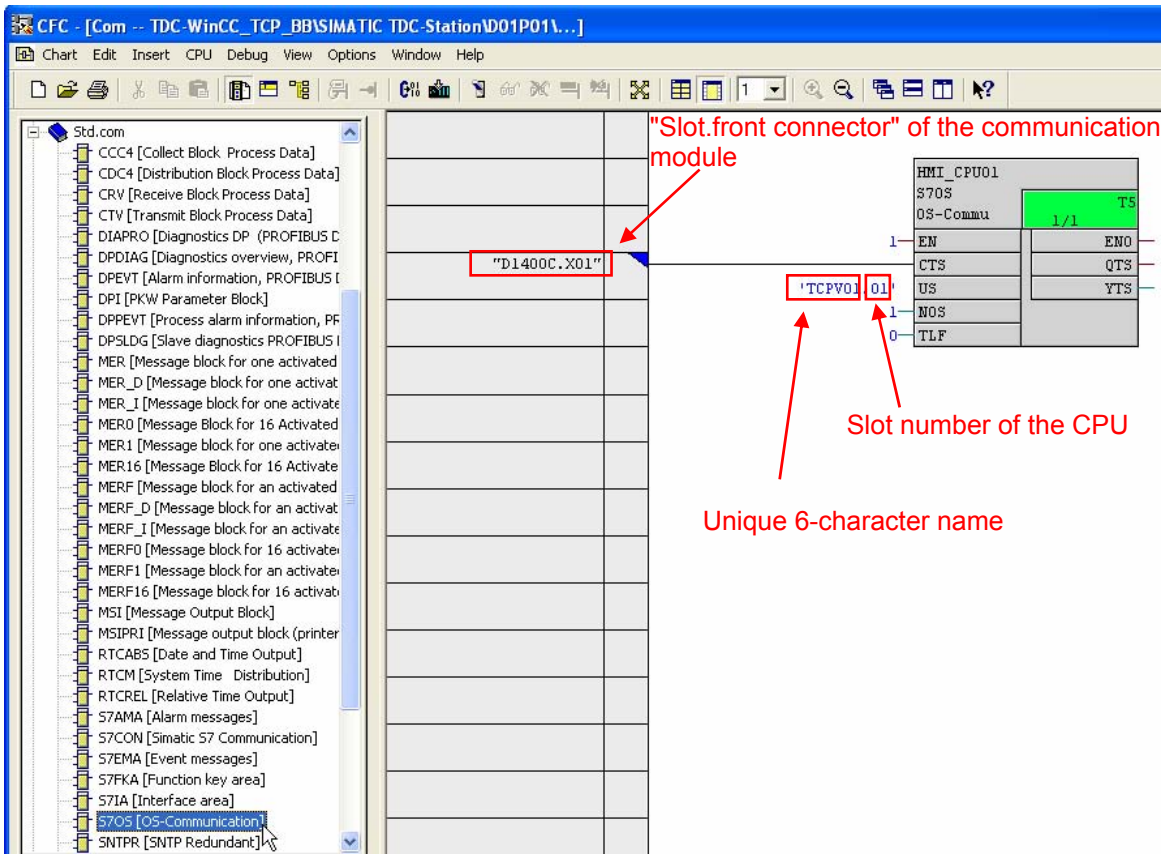
3.16.1.2 Configuring the CFC

3.16.1.2.1 Configuring the coupling-relevant CFC function blocks

Configuring a function block to initialize the TCP/IP interface ("X01") of the CP51M1 module at slot 14 ("D1400C"). The function block should be configured in a task between 32 and 256 ms.



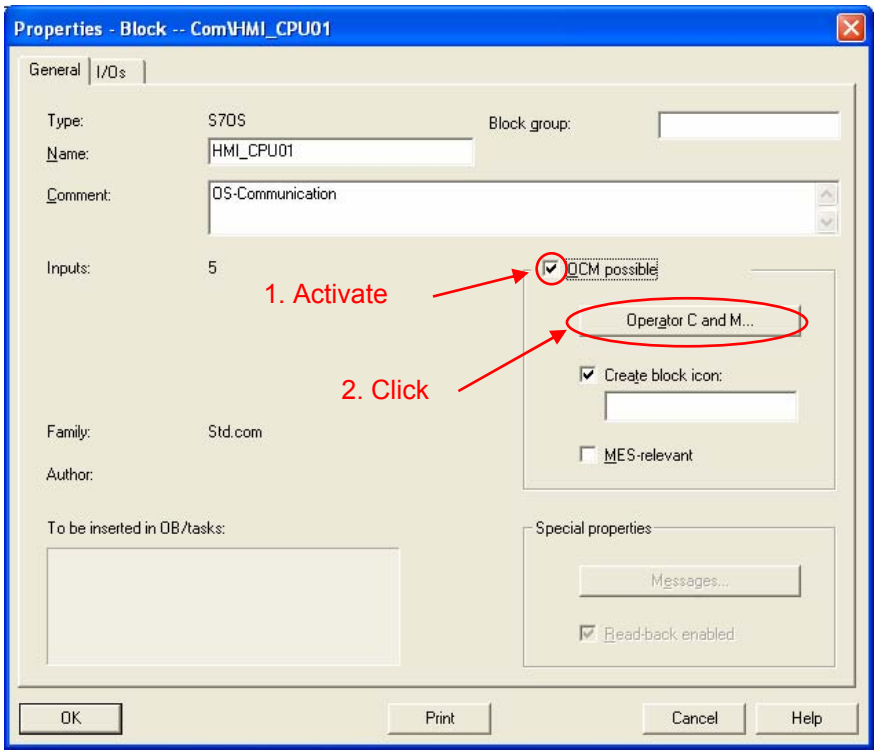
Configuring the "S7OS" function block to initialize the OS communication on the "CPU 01" located in slot 1. Instead of the "S7OS block, block "SER" could also be used. However, this could have a negative influence on the response times. Details regarding this issue can be taken from the description of the function block.



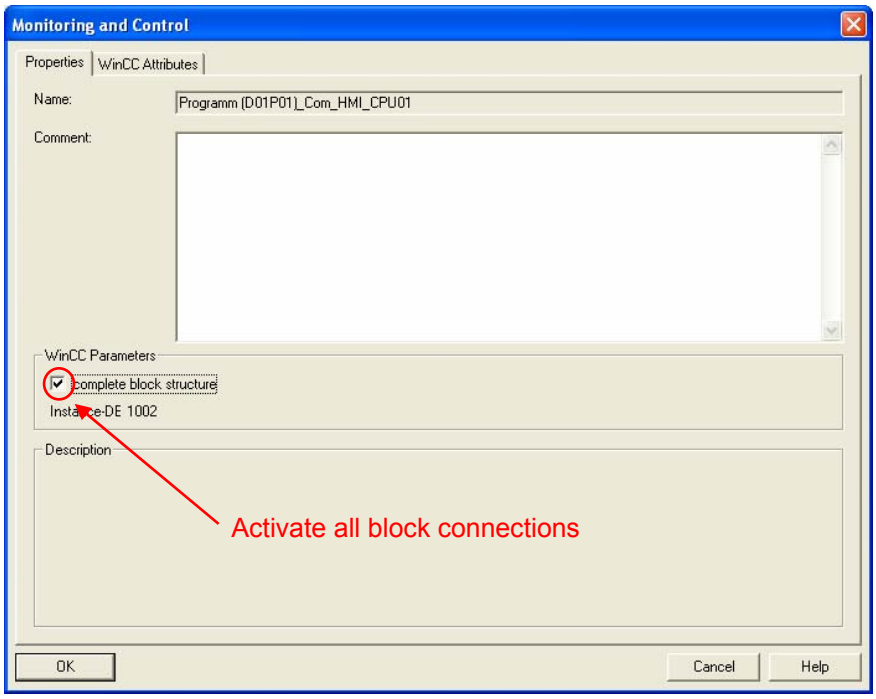
3.16.1.2.2 Marking the function block connections in the CFC charts and creating the address book

The block connections, which must be handled and monitored (OCM) using WinCC must initially be marked in the CFC charts as OCM-capable. To do this, you must proceed in the following steps:

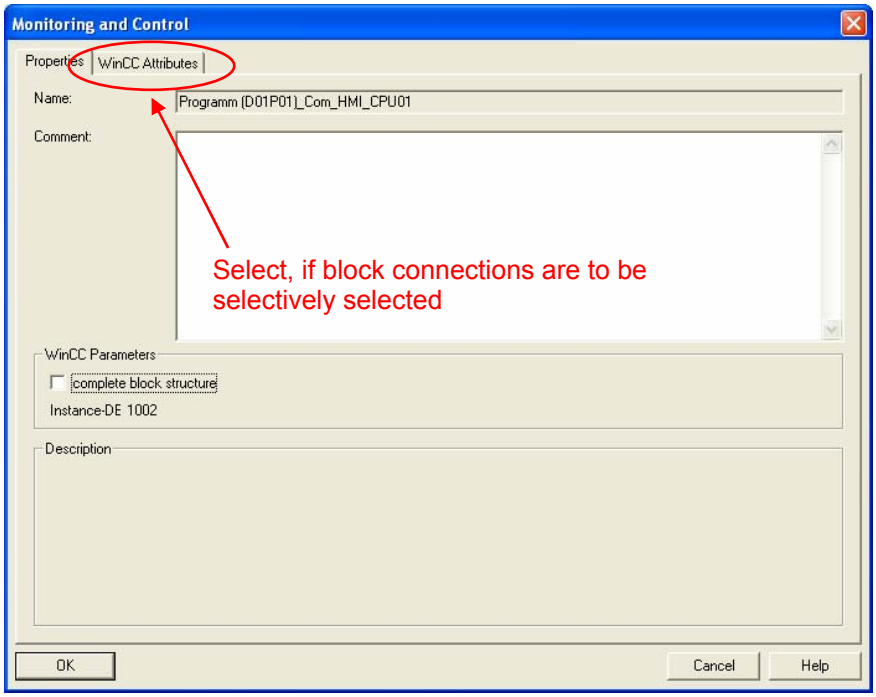
1. Open the Properties dialog box of the block, set the checkmark for "OCM possible" and then press the "Operator C and M" button (refer to the following diagram).

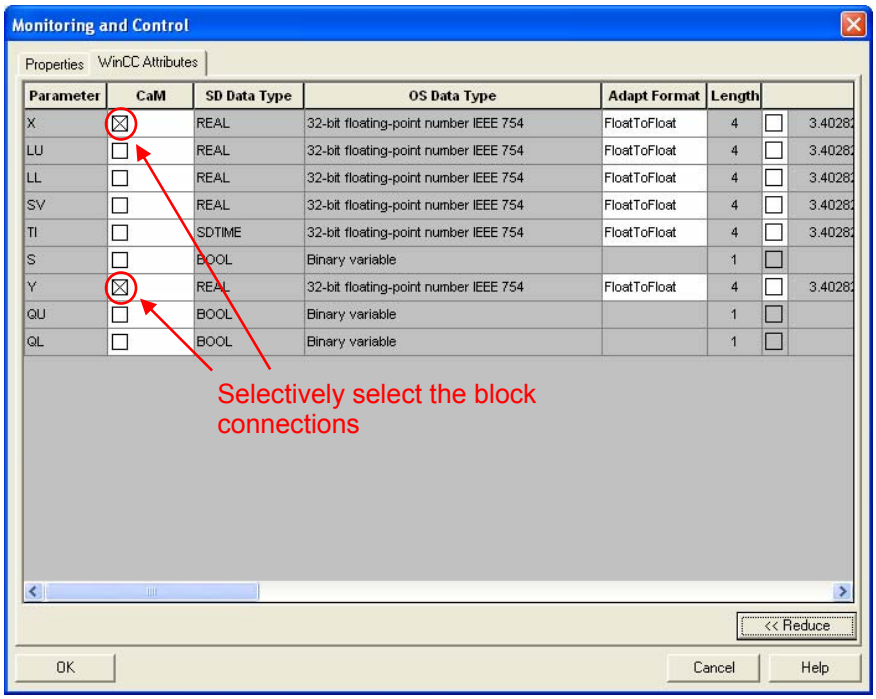


2. In the dialog box that is then displayed, set the checkmark for "Complete block structure", if all connections of the selected block are to be OCM-capable (refer to the following diagram). If only individual connections are to be selected then skip this step and continue with Step 3.



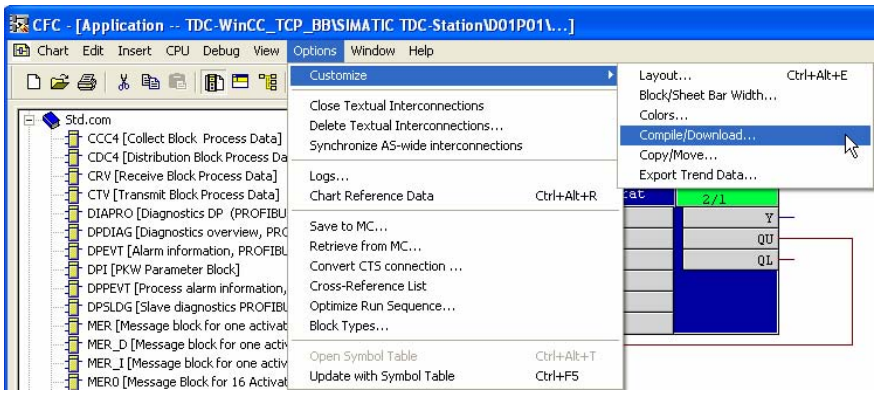
3. Individually mark the OCM-capable connections in the WinCC attribute tab if not all of the connections are to be selected as in Step 2 (refer to the next screenshot).

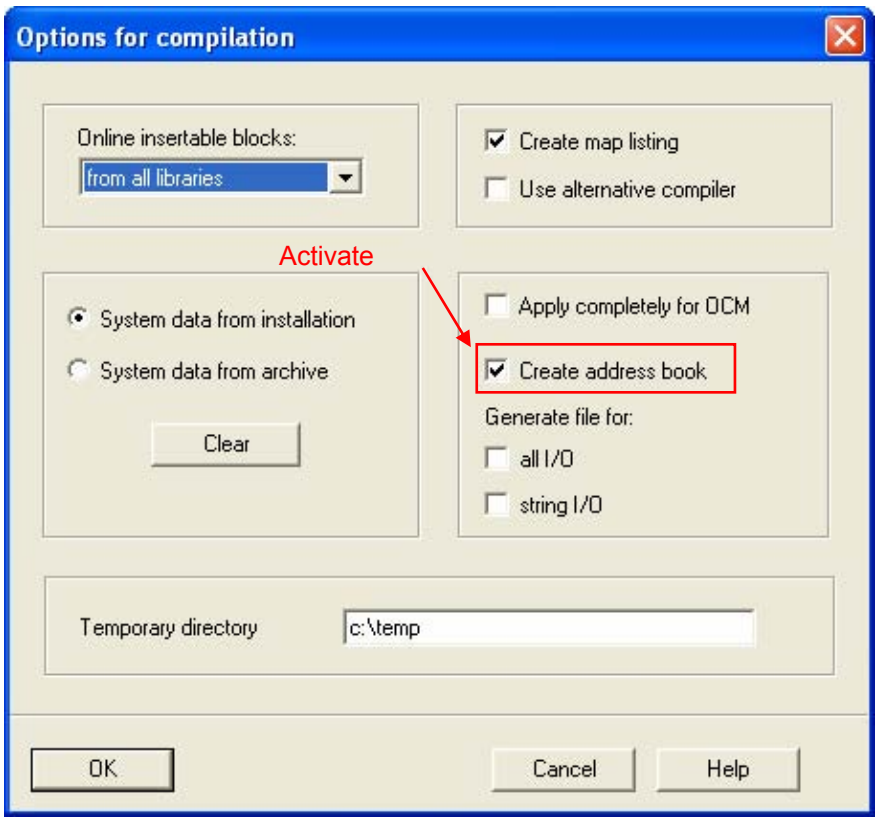




4. Repeat steps 2 and 3 for all of the blocks that are to be controlled and monitored.

5. Select the address book creation in the option dialog box to compile from CFC **(Options→ Customize → Compile/download)** (refer to the next screenshot), in order to obtain the address information for the WinCC configuration.

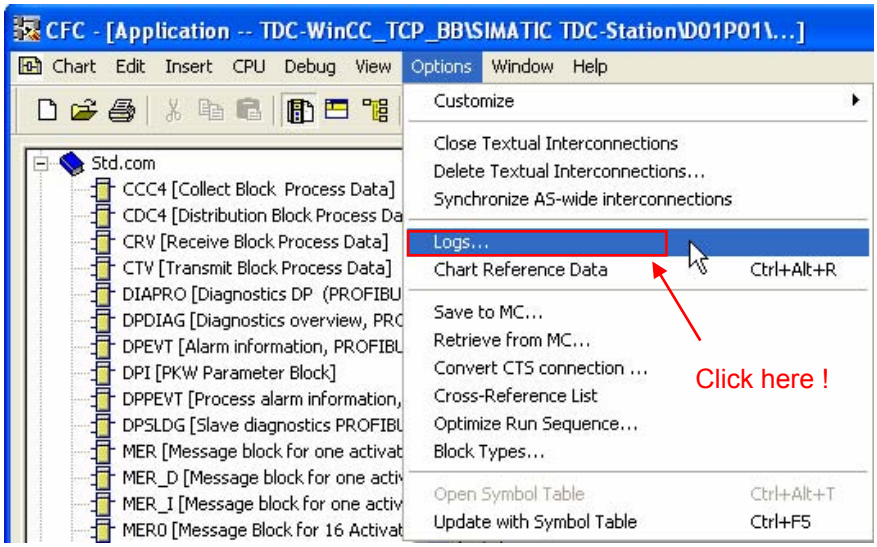




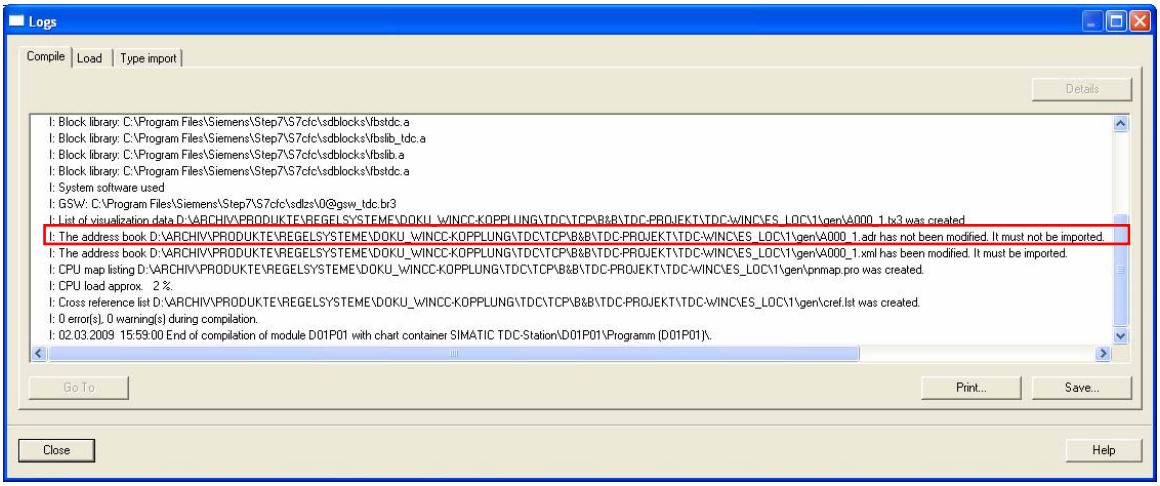
The address book is created when compiling. This then completes all of the activities necessary in the CFC charts. Access using WinCC is possible after it has been downloaded into the target system.

A "D7-SYS-OS engineering tool" – called "Mapper" in the following – is supplied with D7-SYS version 7.1. When run (executed), the tool sets-up a tag (WinCC variable) for each of the function block connectors that was marked.
If this mapper is used, then the workflow can be exited at this point and the work continued in Section 3.16.4 .

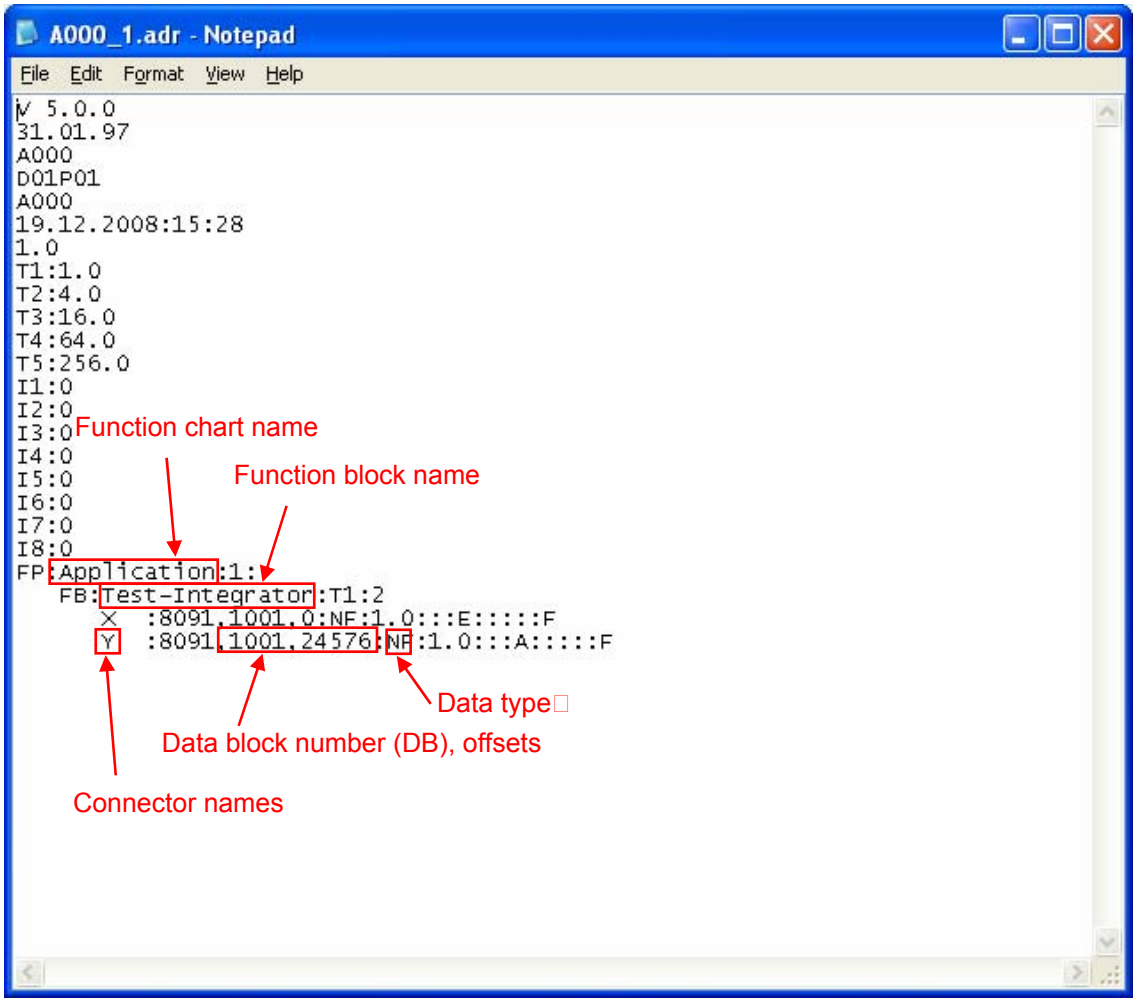
The DB numbers and offsets of the individual connections, necessary to configure WinCC, can now be taken from the address book. A log can be found under the menu item **"Options>Logs"** in which information is provided as to where the address book has been saved.



The location where the address book is saved is marked in the following log diagram:



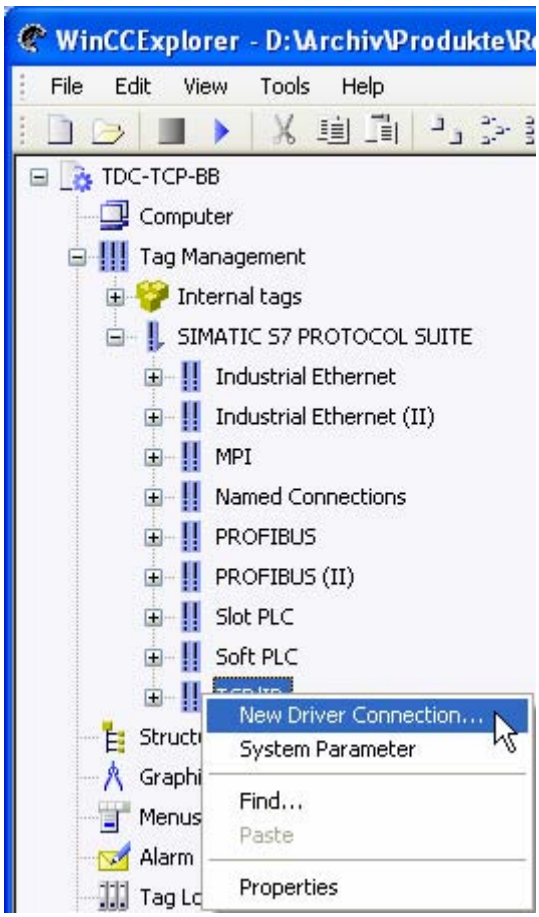
The WinCC-relevant information - such as data block number and offset of the selected function block connectors – can now be taken from the address book.



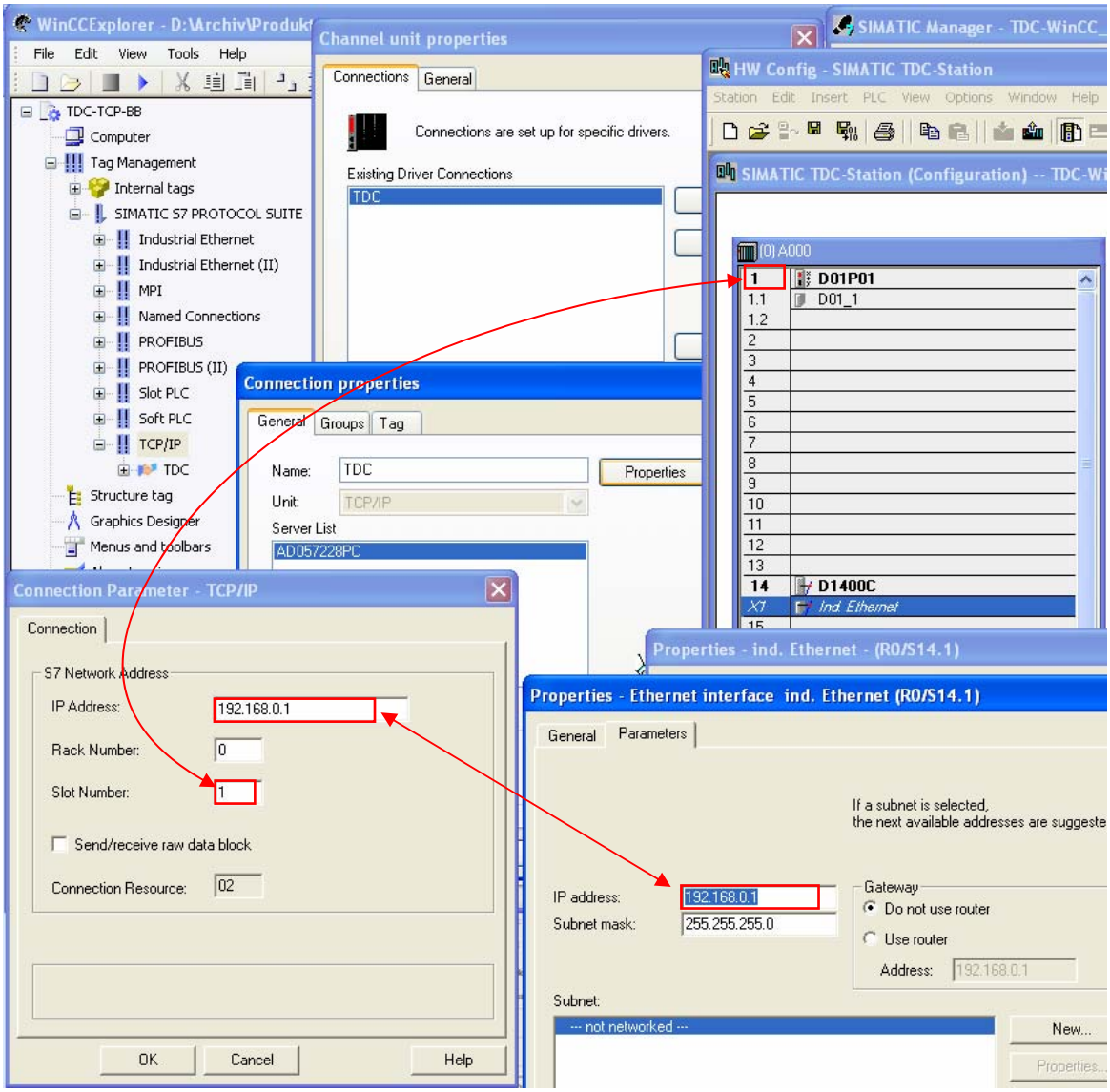
3.16.1.3 Configuring WinCC

When configuring WinCC, proceed in the following steps:

1. Start the WinCC Control Center.
2. Set-up a new project or open an existing one.
3. By selecting **Tag Management → righthand mouse key → Add New Driver → SIMATIC S7 Protocol Suite.CHN → Open** set-up a new driver. If this already exists, then continue with the next step.
4. Set-up a new connection by selecting **TCP/IP → righthand mouse key → New Driver Connection**.

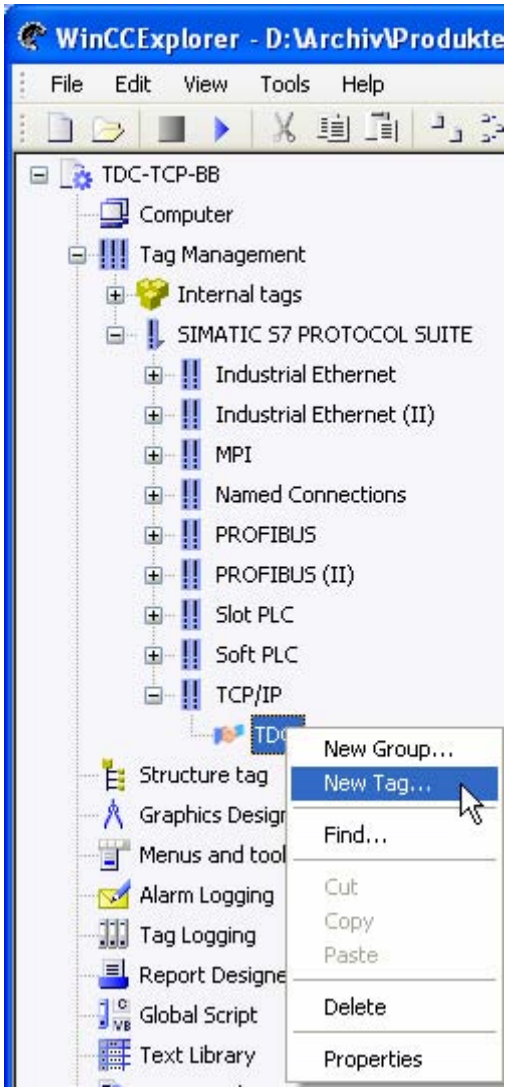


5. To do this, assign a name to the connection in the dialog box, press the properties button and enter the parameter for the connection (TCPI/IP address and slot can be taken from HW Config; refer to the following screenshot).

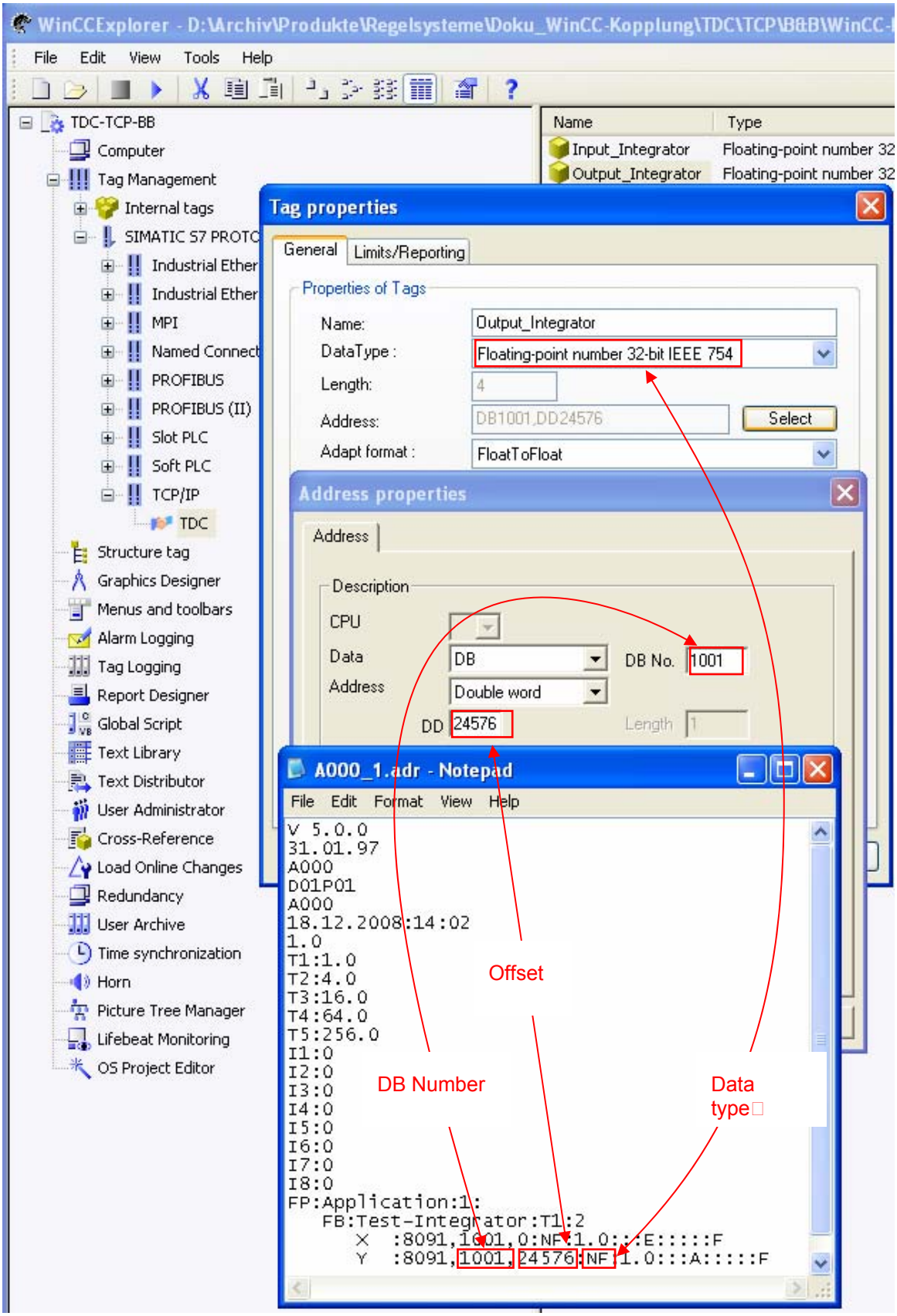


A new TCP/IP connection is created after the dialog box is exited with OK.

6. **Setting-up tags:** Select the connection that has just been set-up using the righthand mouse key and select "New Tag" in the menu that is displayed.

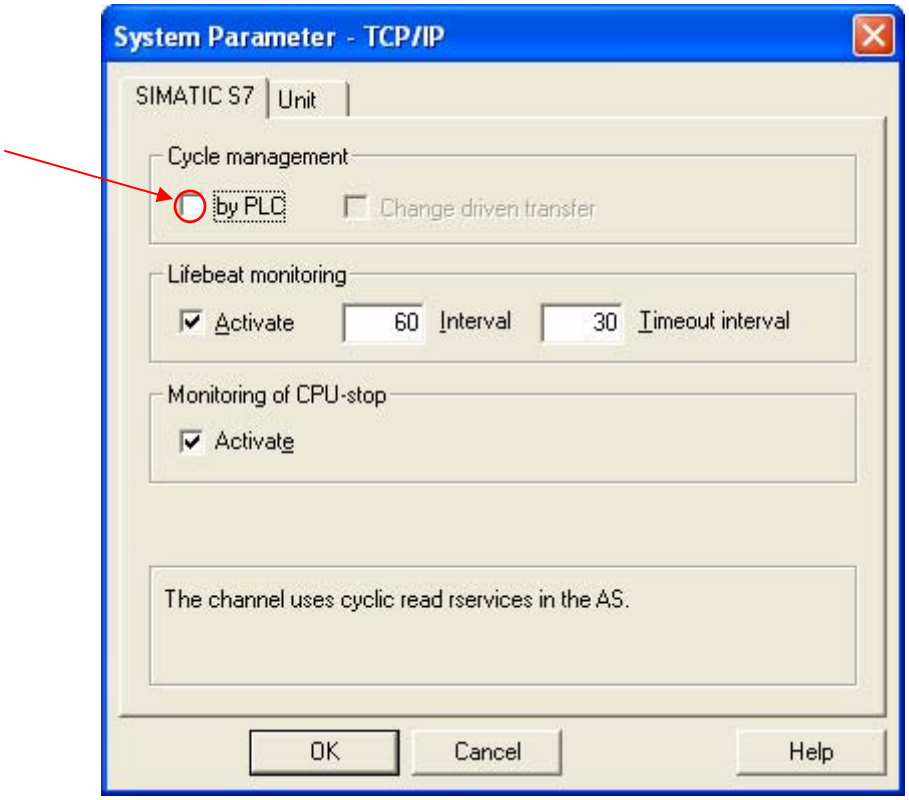


7. In the dialog box that opens, enter a variable name (e.g. function block name_connection name, however, any other name can also be specified).
   The data type of the selected connector, taken from the CFC configuring or from the address book created, can be set under "Data Type" (a reference table of the various data types can be found later on).
   The address dialog box is opened by pressing the "Select" button. The DB number and the offset are specified in this address dialog box. Take this data for the particular connection from the address book created when the CFC was compiled (refer to the next screenshot).

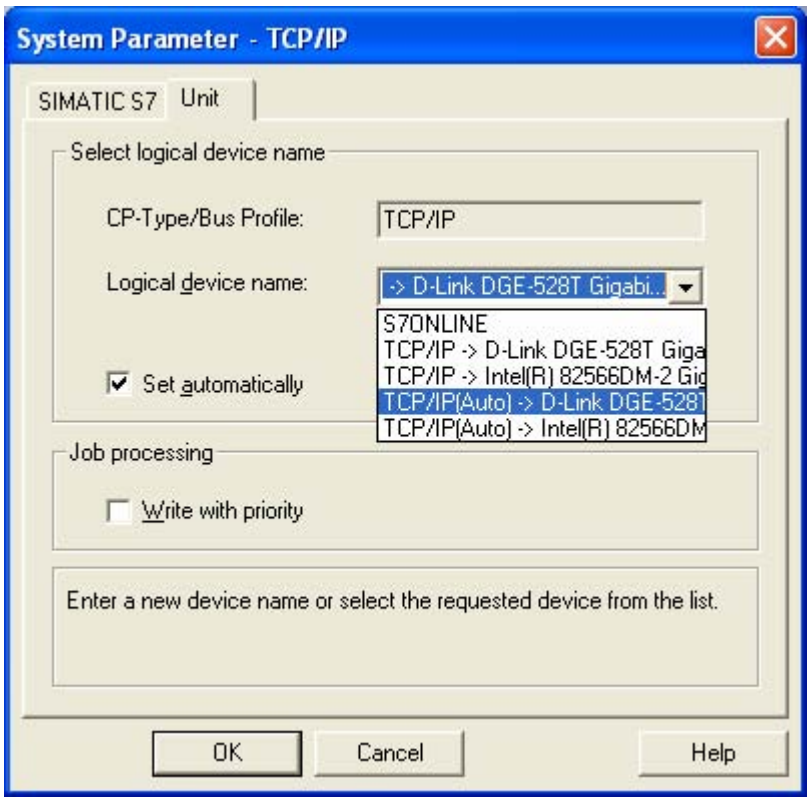

WinCCExplorer - D:\Archiv\Produkte\Regelsysteme\Doku_WinCC-Kopplung\TDC\TCP\B&B\WinCC-
File Edit View Tools Help
TDC-TCP-BB
Computer
Tag Management
Internal tags
SIMATIC S7 PROTO
Industrial Ether
Industrial Ether
MPI
Named Connect
PROFIBUS
PROFIBUS (II)
Slot PLC
Soft PLC
TCP/IP
TDC
Structure tag
Graphics Designer
Menus and toolbars
Alarm Logging
Tag Logging
Report Designer
Global Script
Text Library
Text Distributor
User Administrator
Cross-Reference
Load Online Changes
Redundancy
User Archive
Time synchronization
Horn
Picture Tree Manager
Lifebeat Monitoring
OS Project Editor
Name
Type
Input_Integrator
Floating-point number 32
Output_Integrator
Floating-point number 32
Tag properties
General
Limits/Reporting
Properties of Tags
Name:
Output_Integrator
DataType :
Floating-point number 32-bit IEEE 754
Length:
4
Address:
DB1001,DD24576
Select
Adapt format :
FloatToFloat
Address properties
Address
Description
CPU
Data
DB
DB No.
1001
Address
Double word
DD
24576
Length
1
A000_1.adr - Notepad
File Edit Format View Help
V 5.0.0
31.01.97
A000
D01P01
A000
18.12.2008:14:02
1.0
T1:1.0
T2:4.0
T3:16.0
T4:64.0
T5:256.0
I1:0
I2:0
I3:0
I4:0
I5:0
I6:0
I7:0
I8:0
FP:Application:1:
FB:Test-Integrator:T1:2
X :8091,1001,0:NF:1.0:::E:::::F
Y :8091,1001,24576:NF:1.0:::A:::::F
Offset
DB Number
Data type

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. After entering the appropriate data and exiting the dialog boxes with OK, a variable is set-up in WinCC for the selected block connection.
9. For additional block connections required, the procedure from step 6 onwards should be repeated.
10. In the input window "System Parameter" the driver (TCP / MPI → righthand mouse key → System Parameter) the checkmark for “Cycle management" must **not** be set (refer to the next screenshot).



11. The logical device name has to be select here.



12. Now, a reference can be made in the screen configuring to the tags that have been set-up in this way.

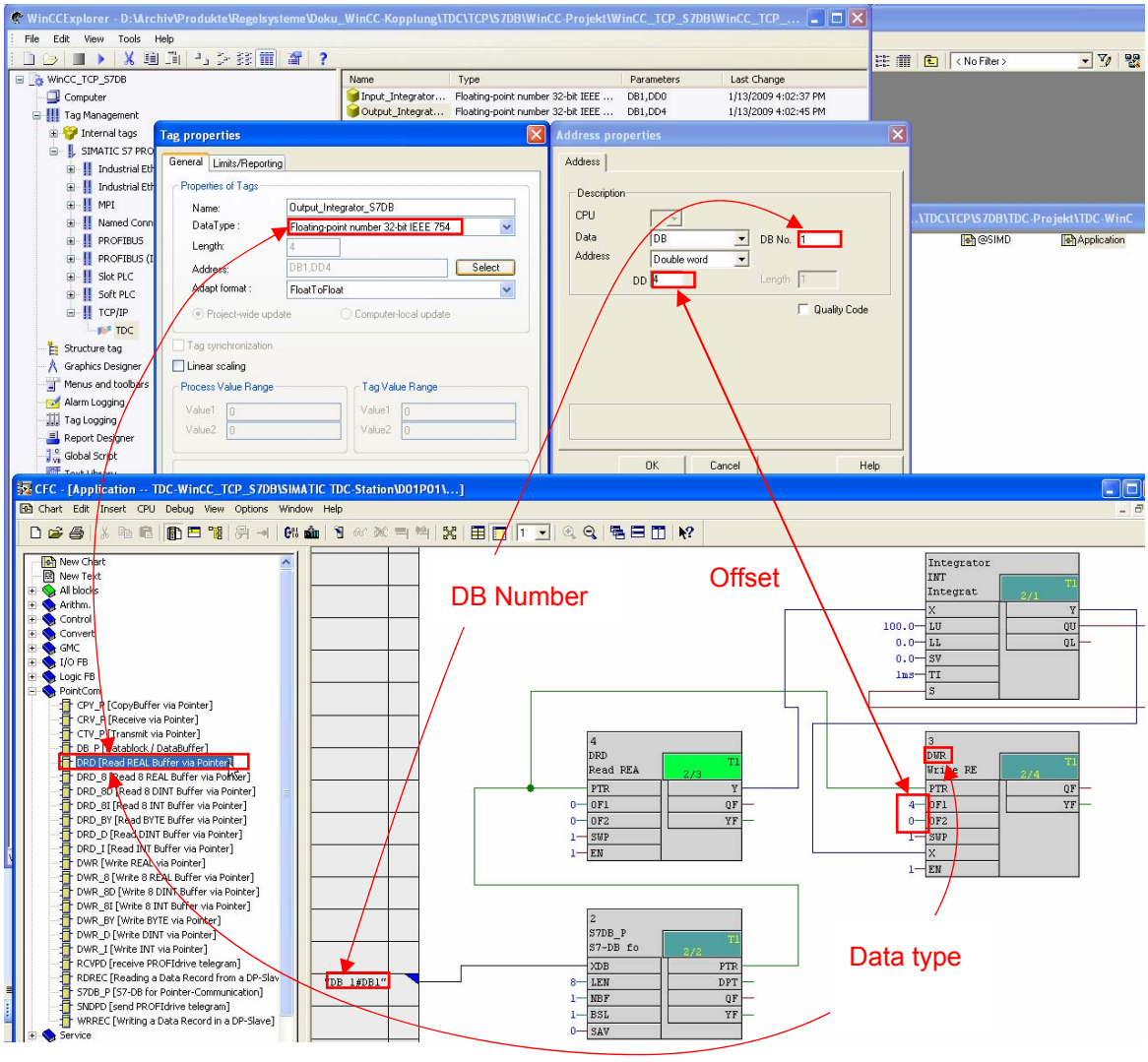
3.16.2 "S7DB" configuration version

When using the "S7DB_P" function block, the connection markings in the CFC charts and the creation of the address book are not required.

Instead of this, the "S7DB_P" function block should be configured with the appropriate pointer-based communication blocks. The block connections, which must be accessed from WinCC, must be connected-up to the S7DB_P function block using "pointer-based communication blocks". Function block S7DB_P sets-up a data block for this data. Please proceed with the following steps:

1. Configure function block S7DB_P. With a righthand mouse click in connector "XDB", select "Interconnection to operands" and specify the required DB number.
2. Connect each connector to be visualized (e.g. as in the next diagram: Connectors "X" and "Y" of the "Integrator" function block) to a dedicated pointer-based communication block from the "pointer com" family of function blocks corresponding to the particular connector type (e.g.: "DRD" function block for read real variables).
3. Connect connector "PTR" of the S7DB_P to the "PTR" connectors of all pointer-based communication blocks.

4. Take the data block number (e.g.: DB1) from the sheet bar and the offset from the pointer-based communication block connector (OF1 + OF2) of the CFC configuring, open the address dialog box by pressing the "Select" button, and then specify the DB number and the offset in this address dialog box (Refer to the next screenshot). Then close the input window with OK.



All of the other configuring steps do not differ from the procedure when using OCM functions.

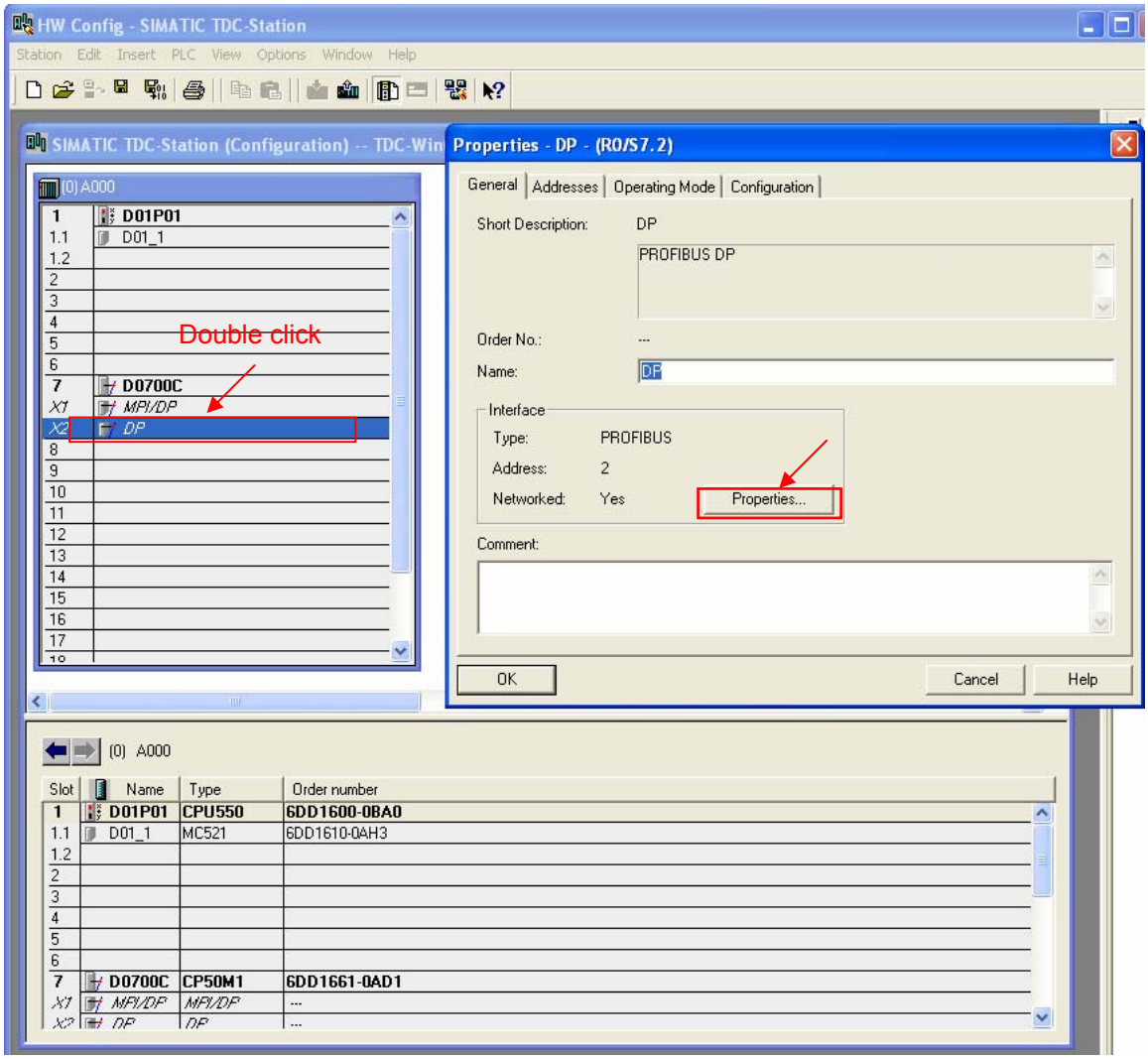
3.16.3 MPI and PROFIBUS DP coupling versions

The deviations relating to either an MPI or DP coupling - when compared to the TCP coupling – that must be taken into account are described in the following chapter.

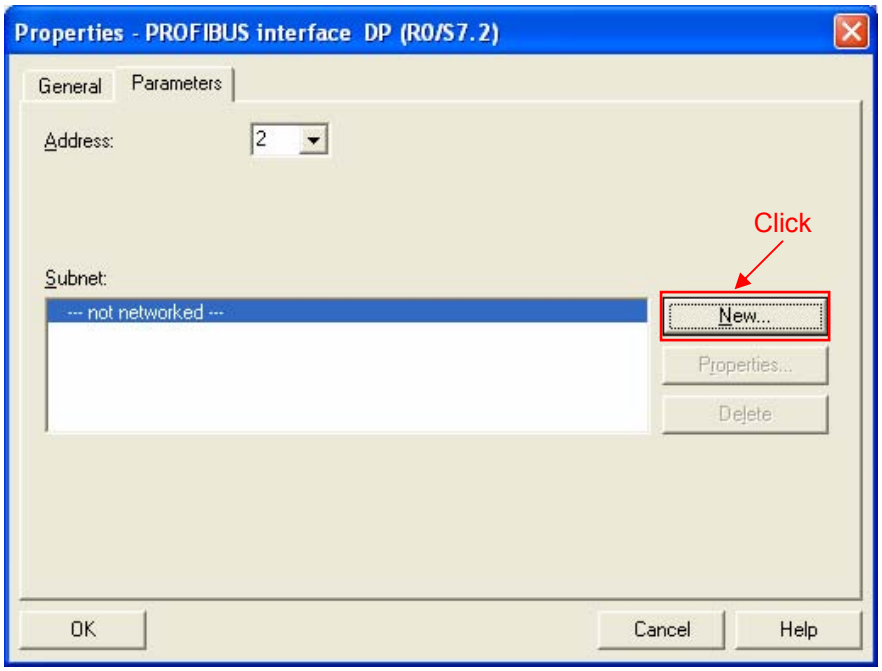
3.16.3.1 Hardware configuration

For a PROFIBUS DP or MPI coupling, as shown in the following example, a CP50M1 is to be configured with the appropriate coupling type.

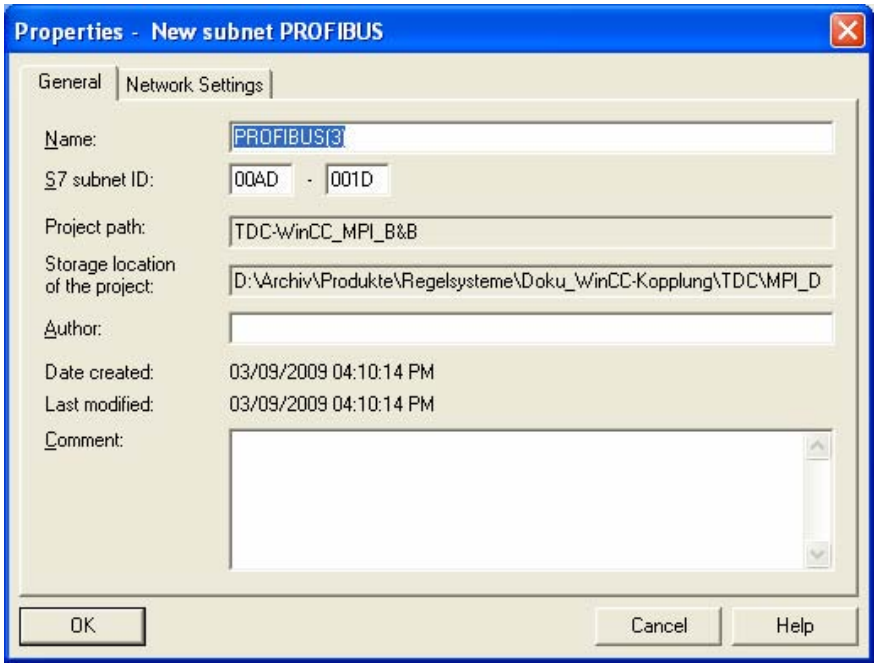
1. Double click on the required CP50M1 interface.
2. Click on "Properties" under the "General" tab.



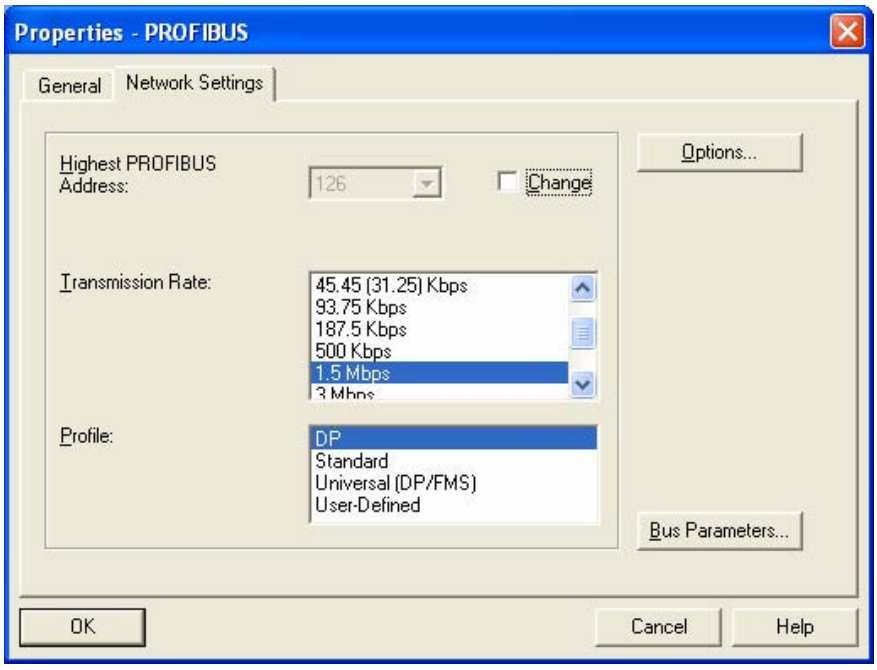
3. In the next window, insert a new "Subnet" by clicking on "New".



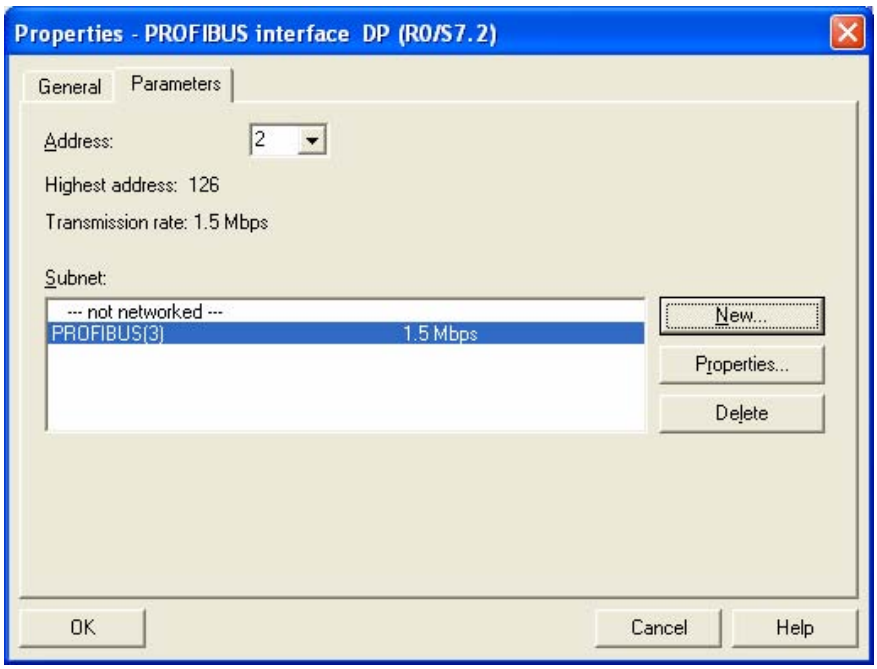
4. If required, change the name and open the "Network Settings" tab.



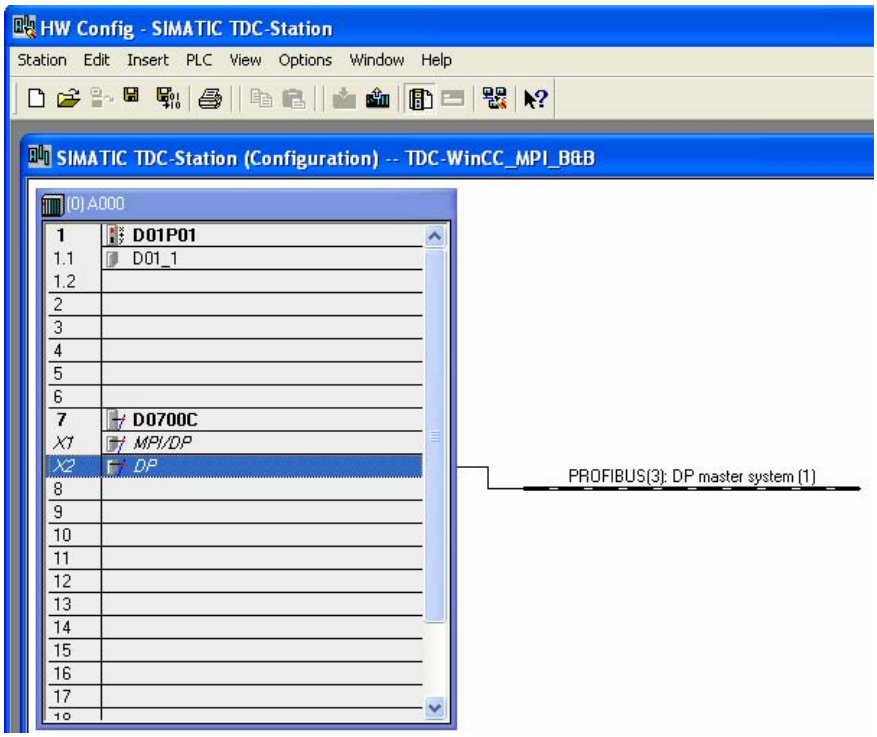
5. Select the required baud rate in the next window.



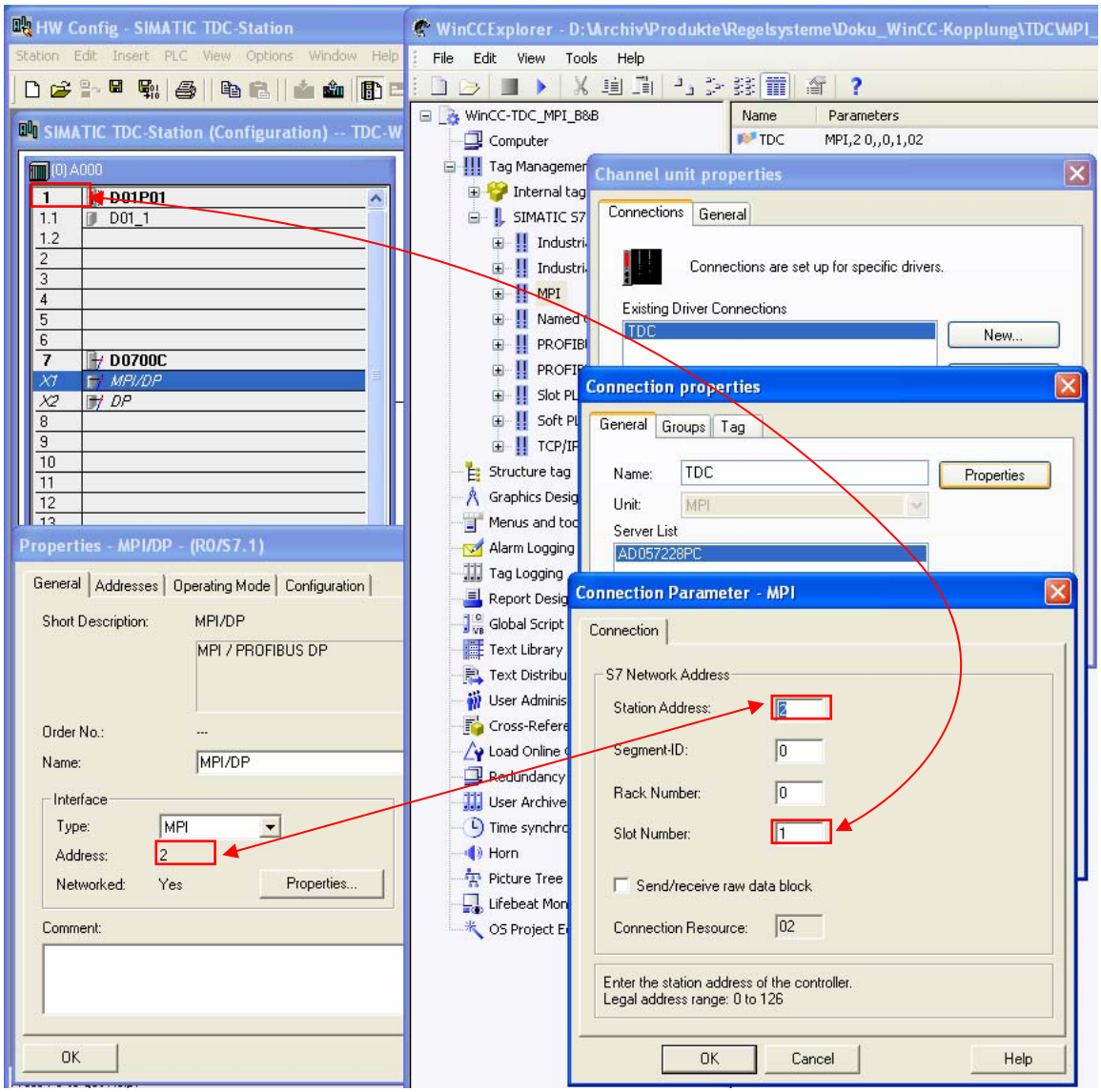
6. Close this window and set the address in the next window. Then close all of the other windows with "OK".



7. This means that the PROFIBUS DP or MPI line is configured as shown in the next screenshot.

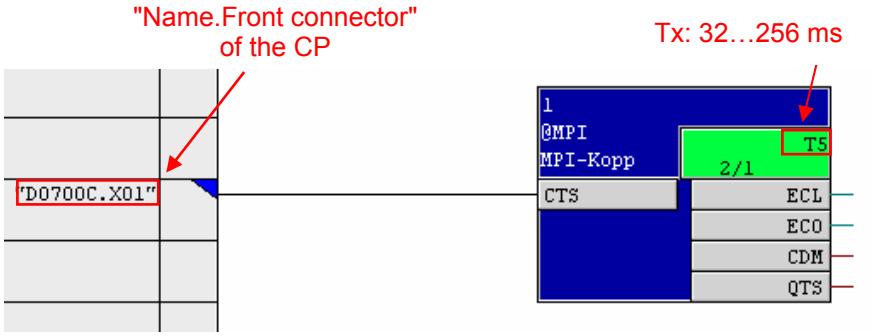


The station address and slot between the TDC station and WinCC should then be aligned as follows:

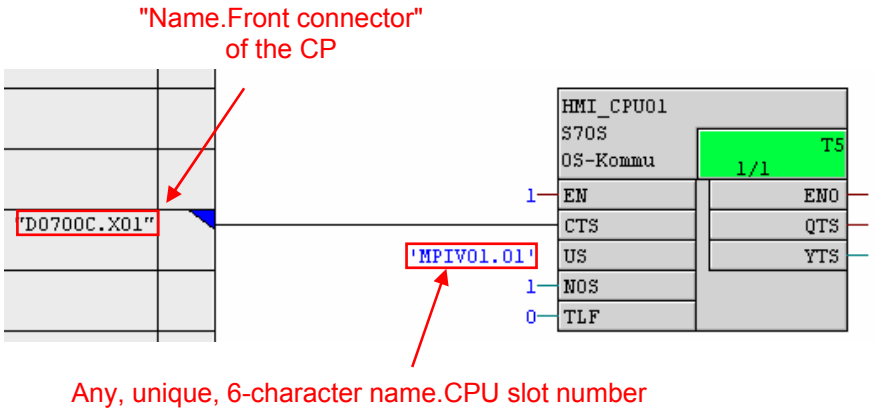


3.16.3.2 Configuring the CFC

To initialize the MPI or DP coupling, the corresponding "@MPI" or "@PRODP" central block should be configured in a task between 32 and 256 ms. The CTS connector should be interconnected to the CP50M1 slot and its front connector.



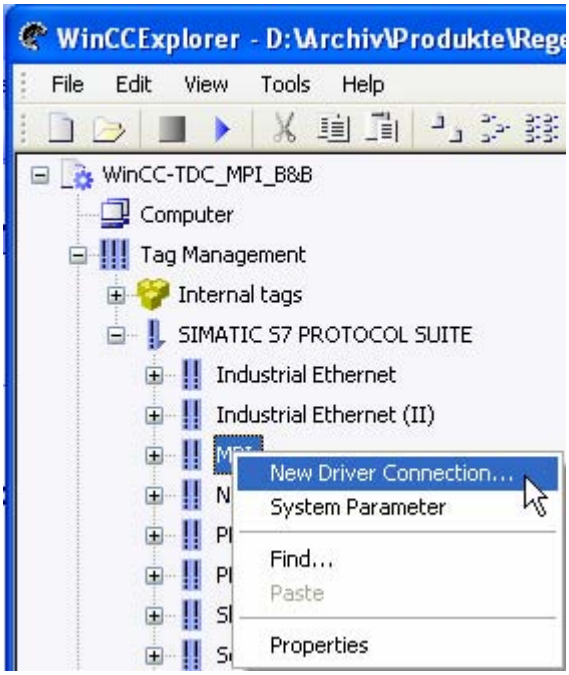
In addition, the "S7OS" function block should also be configured in a task between 32 and 256 ms and interconnected as follows:



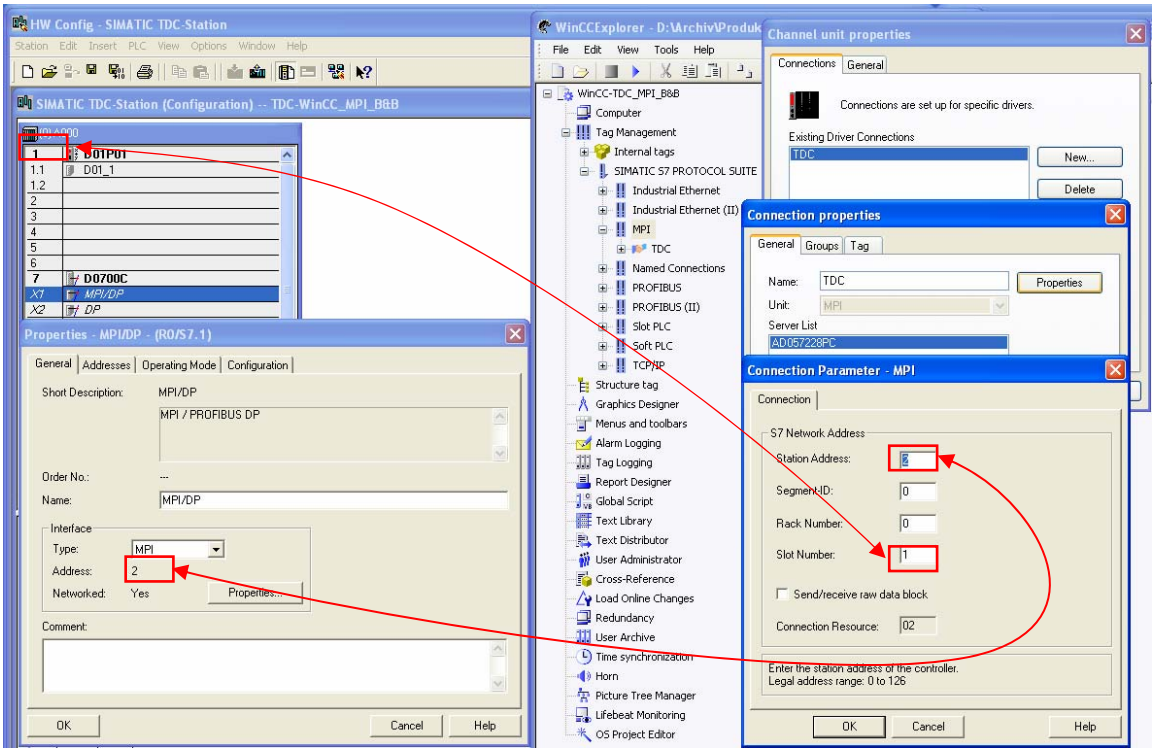
3.16.3.3 Configuring WinCC

1. Set up a new driver by selecting **Tag Management → righthand mouse key → Add New Driver → SIMATIC S7 Protocol Suite.CHN → Open**.
   If this already exists, then continue with the next step.
2. Set-up a new connection by selecting
   MPI → righthand mouse key → New Driver Connection

   If a PROFIBUS DP- coupling is to be used, then this is set-up at this location in the same way
   PROFIBUS DP → righthand mouse key → New Driver Connection.



3. Assign a name in the dialog box, press the Properties button and enter the parameters for the connection.
4. In the WinCC input screen "Connection Parameter" the station address and the slot number - as shown in the following diagram - should be taken from the HW configuration.

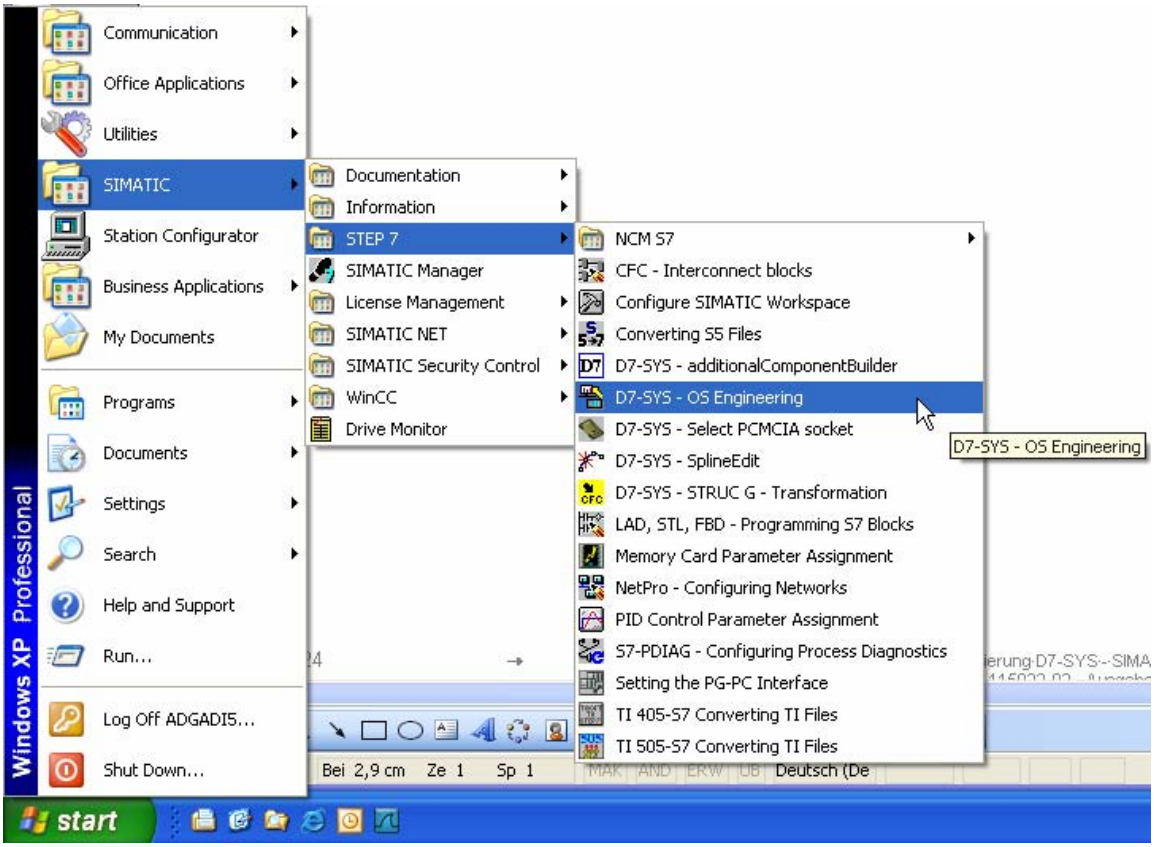


The remaining configuring steps do not differ from those associated with a TCP/IP coupling, and should therefore be taken from the previous chapter.

3.16.4 Configuring using the "D7-SYS-OS engineering tool"

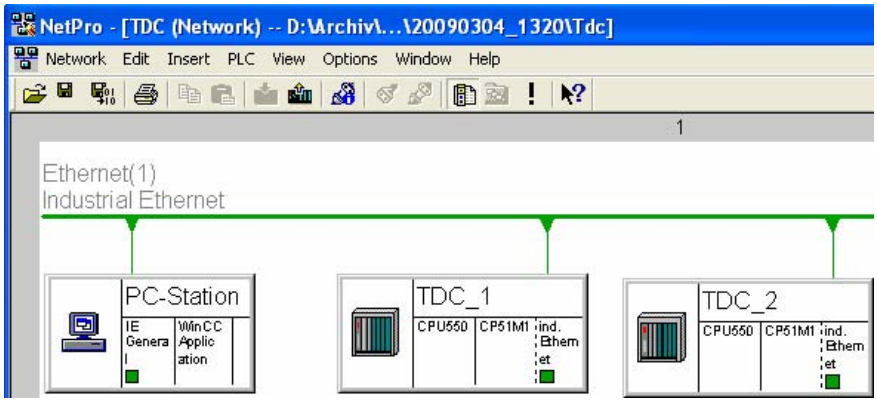
The "D7-SYS engineering tool" - also called "Mapper" in the following text - sets-up tags for the selected connectors of the CFC function blocks. These tags can then be further processed by WinCC.
The following chapter describes how to use this tool.

Calling the mapper:

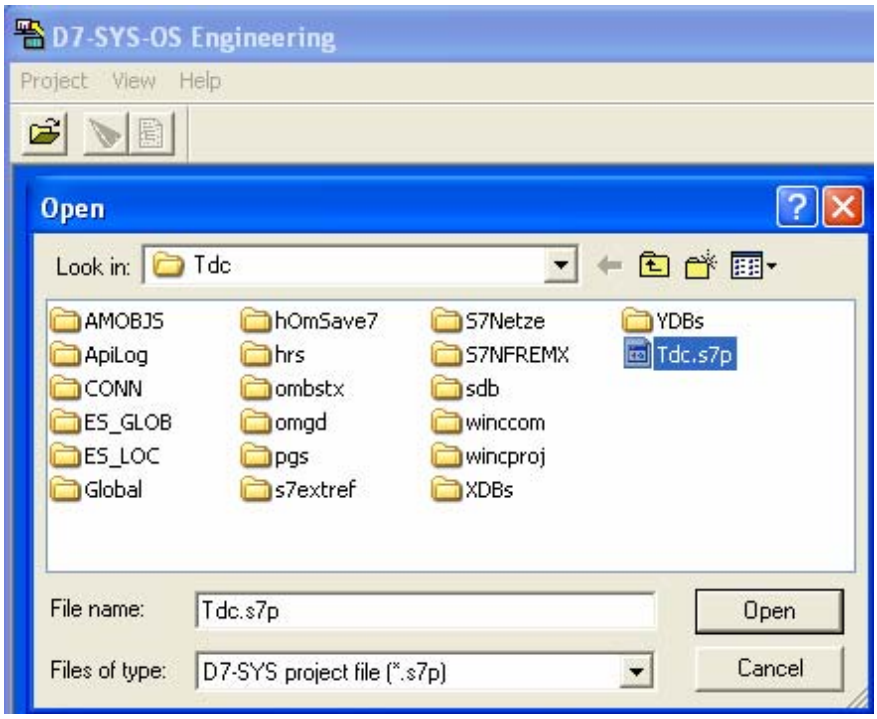


Prerequisites for mapping:

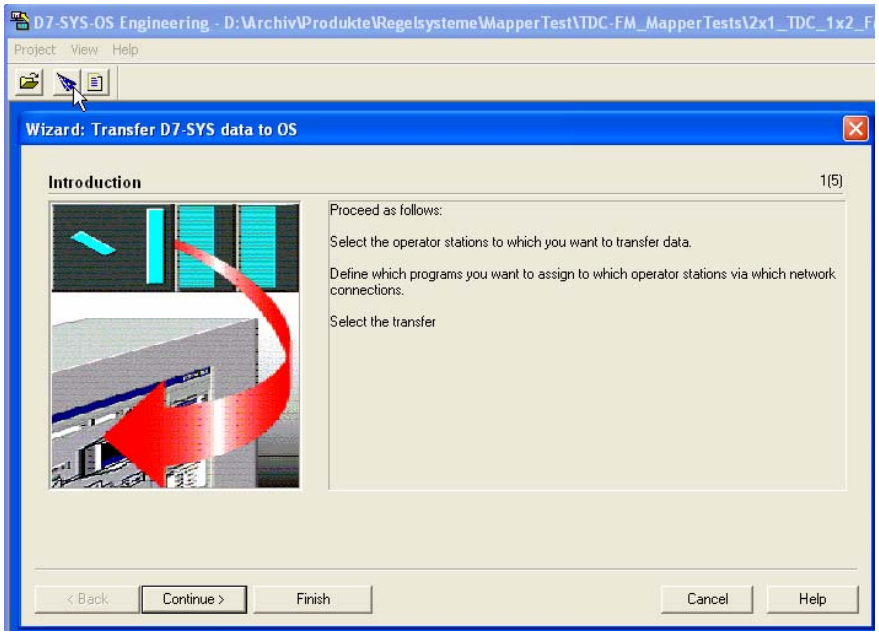
- Select the CFC function block connectors and then compile with the "Create address book" option activated, as described in the previous chapter.
- Inserted PC station with communication module and WinCC application.
- The "NetPro" configuration should be checked as to whether all stations are connected with one another via the required coupling types; for example, in this case, via TCP/IP.
  The same procedure should be used for MPI or DP couplings.
- A transfer to WinCC can only proceed, if the WinCC explorer has been started and the related WinCC project has been loaded.



After the call, the target project should be selected using the lefthand "Open" icon.

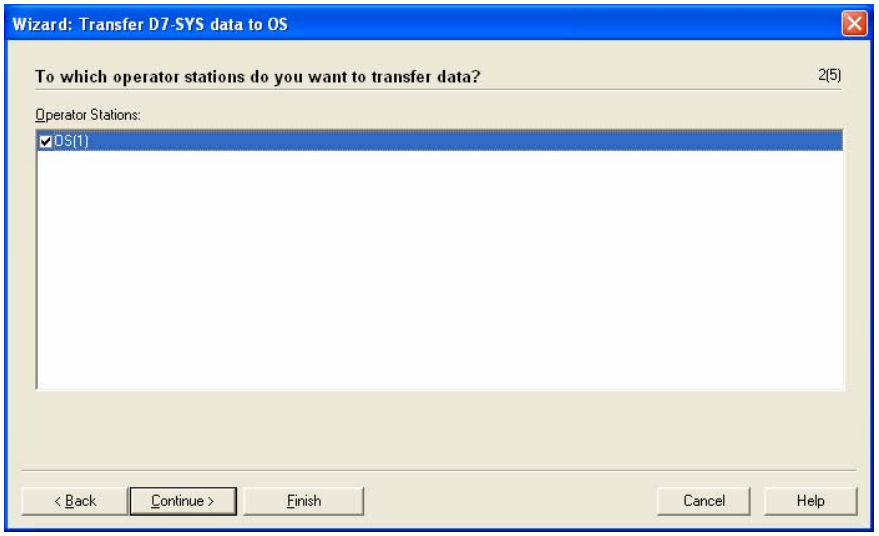


A Wizard is started by clicking twice on the 2nd icon (Wizard's hat):



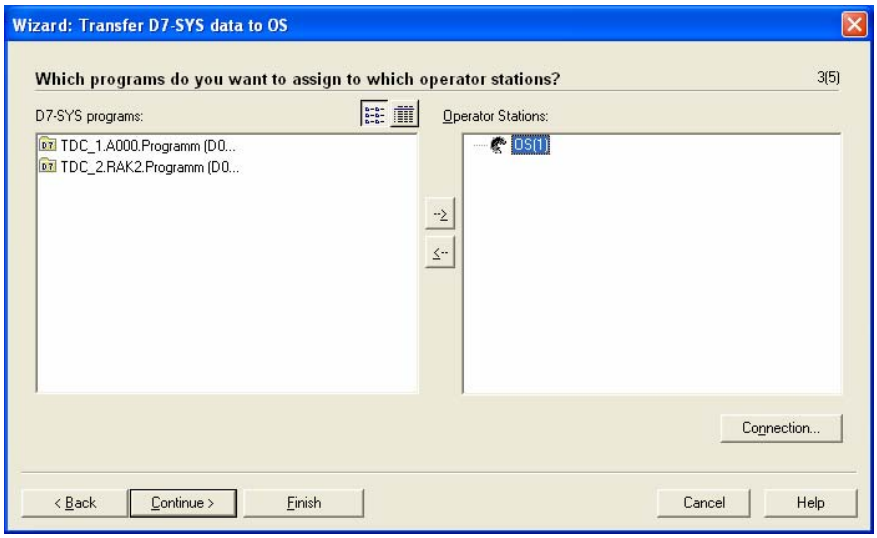
Then click on "Continue":

The operator station is then selected as shown in the next screenshot:

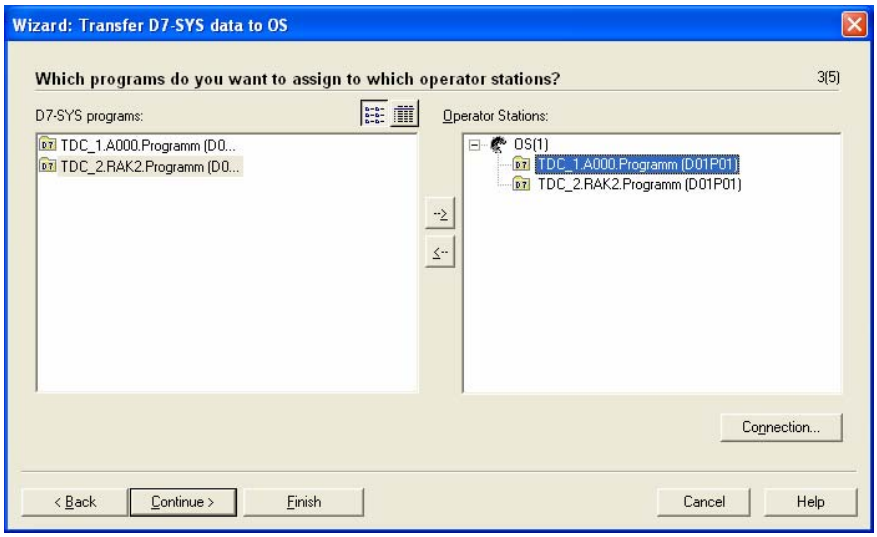


Then click on "Continue":

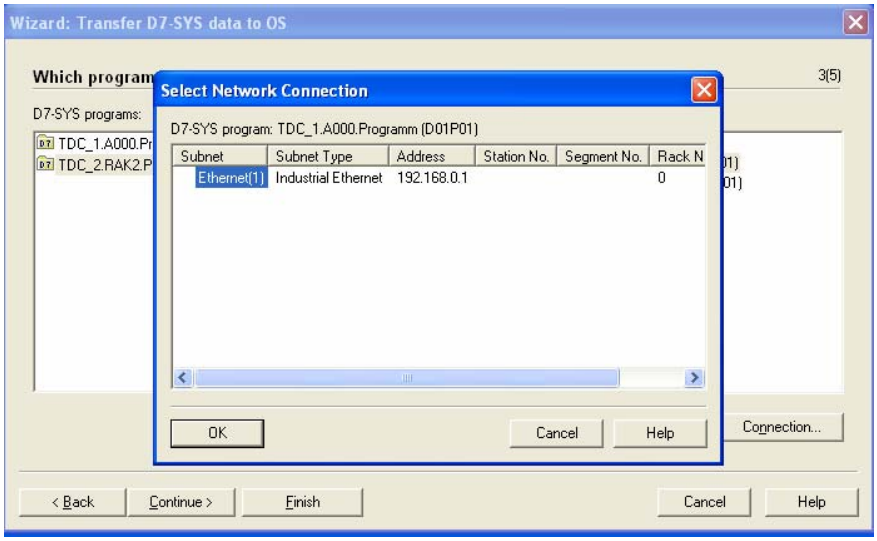
The selection and assignment of the programs to the operator stations is then realized in the next step:



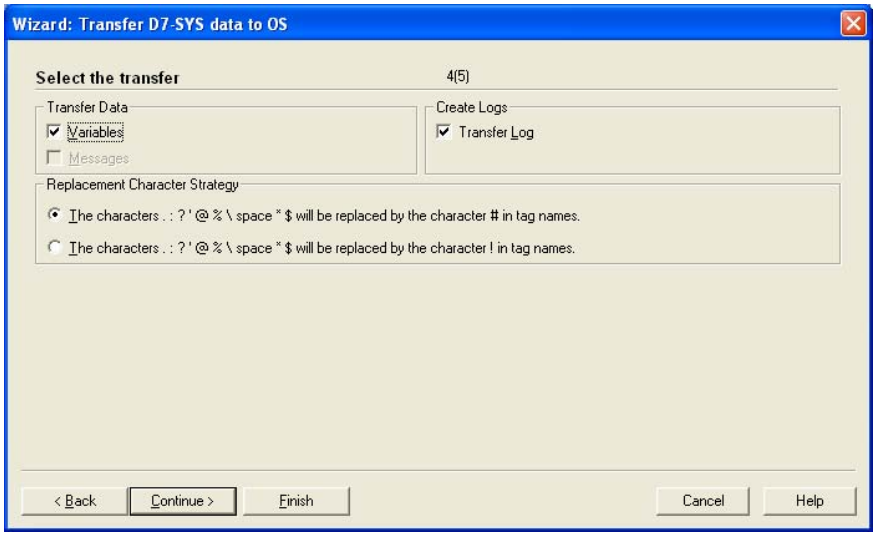
Displaying the selected programs:



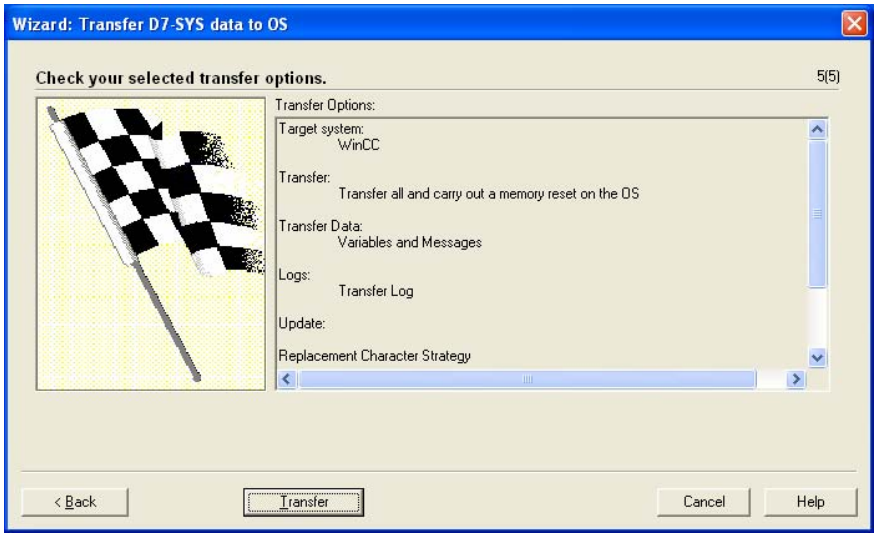
If the connection parameters are to be checked, then click on "Connection". This is not absolutely necessary if "NetPro" has been correctly configured. However, if various couplings are used (TCP/IP,MPI or DP) then the required connection should be selected.



Then click on "Continue":

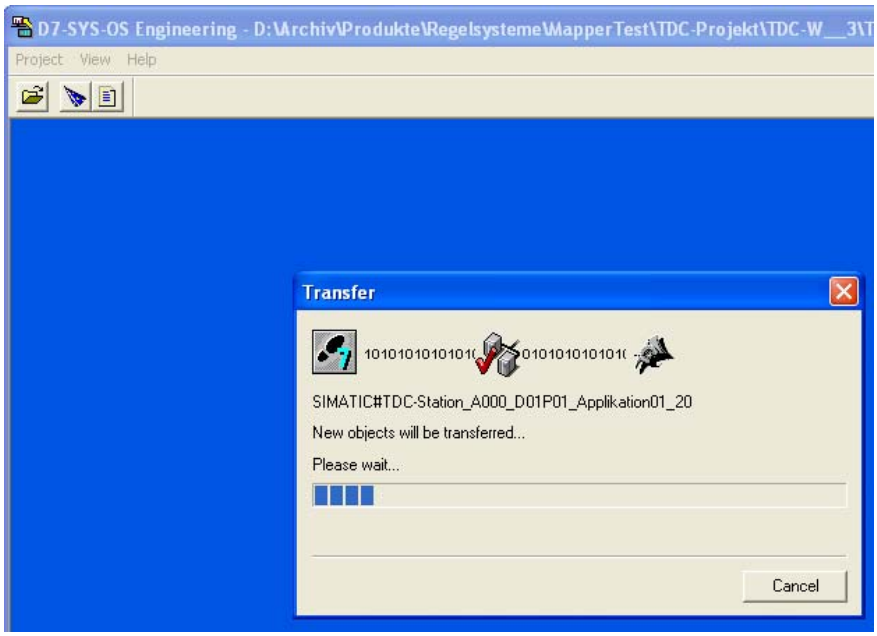


Then click on "Continue":



Then, start the actual mapping (transfer) process by clicking on "Transfer":

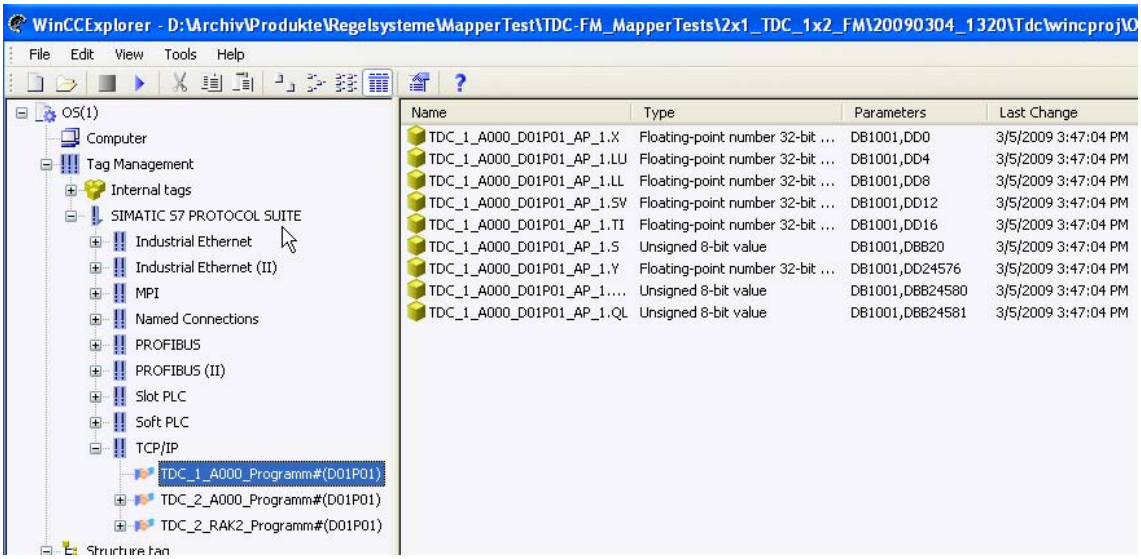
Transferring (mapping) then runs:



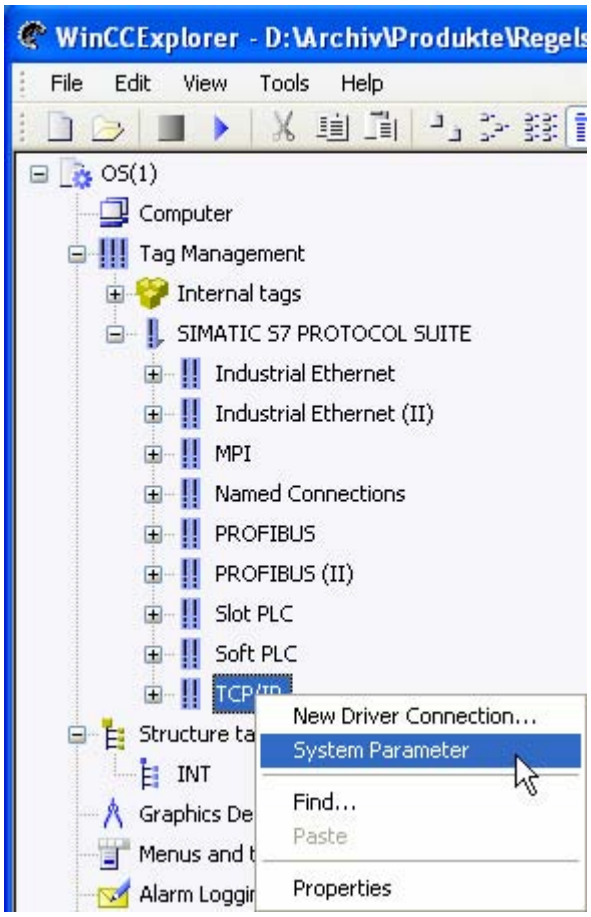
This therefore completes data transfer:

End by clicking on "OK".

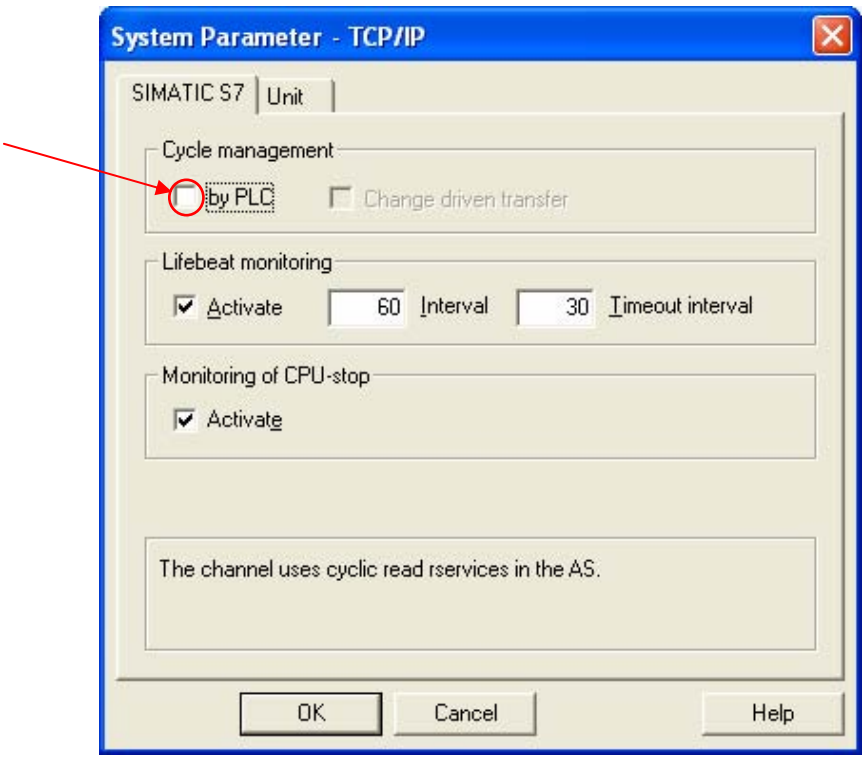
The tags that have been created are listed in WinCC Explorer:



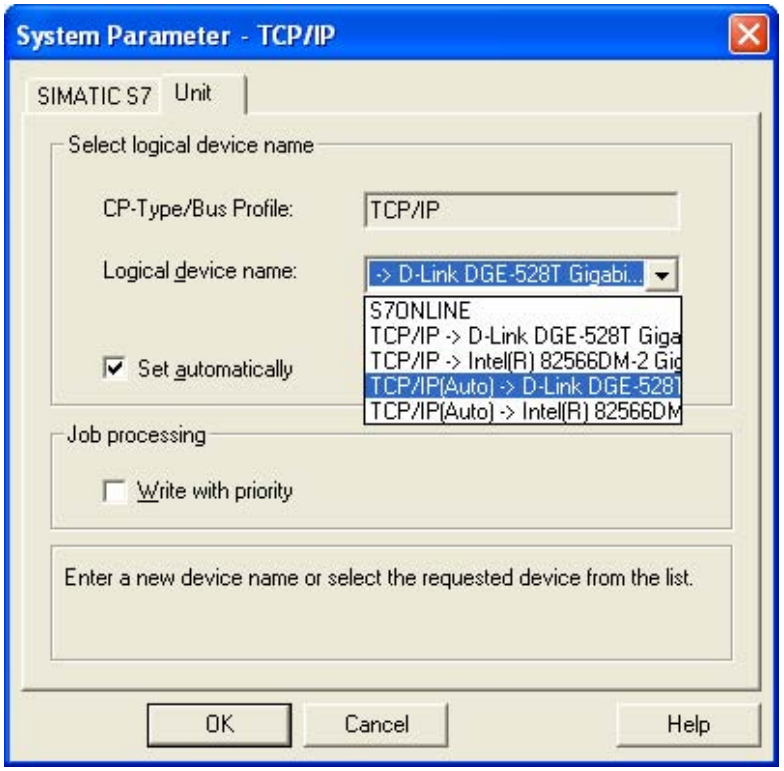
The system parameters of the coupling, shown in the following diagrams, now have to be checked:



Deactivate cycle management by the automation system:



Select the device name:



After changing the system parameters WinCC has to be restarted. It will be possible to access the configured variables once this has been completed.

3.17 Communications with WinCC (TCP/IP)

Introduction

This User Manual shows you how you can couple WinCC to SIMATIC TDC via a TCP/IP coupling using a simple example of the configuring software. All of the necessary configuring steps (including the hardware- and software requirements) are described. The handling of the necessary software tools is not described here, but a reference is made to the appropriate User Manuals.

3.17.1 Prerequisites

Software

TDC PMC TCP channel-DLL	**Software prerequisites** • WinCC-Systemsoftware: fom version 5.0 für Windows NT 4.0 • SIMATIC TDC: PCS7 fom version 5.0 mit D7-SYS V5.1 **Order No. (SIMATIC TDC PMC TCP/IP)** 2XV9450-1WC43-0KX0 **Further information** Siemens AG Industrial Solutions and Services IT Plant Solutions I&S IT PS 3 Werner-von-Siemens-Straße, 60 91052 Erlangen Contact: your IT4Industry Team E-Mail: info@it4industry.de WWW: http://www. it4industry.de
Tools	**PROBI:** The configuring package PROBI is component of every SIMADYN D-PMC licence.

Table 3-52 Software prerequisites

NOTE The TDC PMC TCP channel-DLL can only be used in conjunction with WinCC from V5.0. The software is installed via a setup routine, which is provided on the product software floppy disk.

Hardware PC configuring station:

network card for TCP/IP, for example 3COM

Table 3-53 Hardware prerequisites

SIMATIC TDC hardware

System:	D7-SYS from V5.1	
Subrack:	UR5213	21 slots with fan
Slot 1:	CPU550	CPU (with local service interface)
Slot 1.1:	MC500	4 Mbyte Flash memory module
Slot 2:	CP50M1/CP50M0	Communications buffer module
Slot 18:	CP51M1/CP5100	SINEC TCP/IP interface

Table 3-54 Hardware design for the configuring example

3.17.2 Process variables

A SIMATIC TDC station must be configured and parameterized and a test chart generated using the CFC *configuring tool. The hardware configuration is described under* ***Point 26.1 (*****SIMATIC TDC hardware design)**. We will not discuss in detail here how SIMATIC TDC software is generated using the CFC. If you require further information refer to the Configuring Instructions.

3.17.2.1 SIMATIC TDC software

The CFC chart for the WinCC link does not have to be realized on a separate chart, but is however recommended as this is more transparent. The following function blocks are required for the coupling between SIMATIC TDC and WinCC for process variables:

- LI - LAN interface block
- VM visualization block
- VI interface block
- VC concentrator block
- CI interface block
- SER02 communications block

The blocks are connected as follows:

(Only the relevant I/O are described)

FB LI

I/O name	Significance	Example
CTS	Name of the interface used (CP51M1/CP5100)	D1800C
AT		
AR	[channel name].[protocol]-[local port-no	ARC01.T-19001 with T for TCP/IP
NA	Maximum number of parallel jobs from WinCC	20
NC	WinCC ID	0
COM	Communications medium TCP/IP	3
CCV	Connection with VM, connection CVP	<VM.CVP
CCF	No connection with the FM block	16#0
CCB	No connection with the MM block	16#0

FB VM

I/O name	Significance	Example
NA	Sum of the jobs reserved for the VM	20
NL	No. of LI blocks	1
NV	No. of VI blocks	1
MEM	Default	0
TGL	Default	0
CVP	Connection with LI.CCV,VI.CCV	>(LI.CCV,VI.CCV)

FB VI

I/O name	Significance	Example
CTS	Processor name	D01P01
AT	Send channel name to the VC	´CMDVCH`
AR	Receive channel name from the VC	ACKVCH´
CCV	Connection with VM, connection CVP	´VM.CVP´

FB VC

I/O name	Significance	Example
CTS	Processor name	D01P01
AT	Send channel name to the VI	´ACKVCH´
AR	Receive channel name from the VI	´CMDVCH´
NC	No. of connected CIs	8
CVP	Connection with CI, connection CCV	>(CI.CCV)

FB CI

I/O name	Significance	Example
CTS	Processor name	D01P01
AT	Send channel name to the SER02	´CMDH´
AR	Receive channel name from the SER02	´ACKH´
ADT	Data channel name from the SER02	´DATH´
CCV	Connection with VC, connection CVP	<VC.CVP

FB SER02

I/O name	Significance	Example
CTS	Processor name	D01P01
AT	Send channel name to the CI	´ACKH´
AR	Receive channel name from the CI	´CMDH´
ADT	Data channel name to the CI	´DATH´
CLT	Length, send channel	116
CLR	Length, receive channel	524
CLD	Length, data channel	432
TPD	For operator control and visualization (HMI)	0
NL	Maximum number of MWLs (measured value lists)	50
NV	Max. number of measured values (connections)	1000

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 @TCPIP

I/O name	Significance	Example
CTS	Name of the interface used (CP51M1/CP5100)	´D1800C´

FB @GLOB

I/O name	Significance	Example
CTS	Buffer memory module name	´D0200A´
CDV	Memory re-structure (1)	0

3.17.2.2 Configuring WinCC

For the particular example, it is sufficient to use a basic WinCC configuring software with several input/output fields.

We will not discuss the WinCC configuring software here. If you require further information refer to the comprehensive WinCC Configuring Manuals. We recommend the Getting Started SIMATIC WinCC Manual for an introduction into configuring WinCC.

3.17.3 Binary events

SIMATIC TDC configuring software

No additional configuring is required for process value visualization for the binary event technique with WinCC. The selection regarding which bit of a variable initiates which message is realized exclusively in WinCC. The configuring rules to output process variables remain.

WinCC configuring software

In addition to configuring software for the process variables, an ALARM logging configuring software must be generated. We will not discuss the WinCC configuring software here. If you require more detailed information, refer to the comprehensive WinCC Configuring Manuals. We recommend the Getting Started SIMATIC WinCC Manual for an introduction into configuring WinCC.

3.17.4 SIMATIC TDC messages

3.17.4.1 SIMATIC TDC configuring software

To output messages from SIMATIC TDC to WinCC, the WinCC block MM is required in addition to configuring the process value output :

- MM message manager

The blocks are connected as follows :

(Only the relevant connections are described)

FB MM

I/O name	Significance	Example
CTS	Processor name	D01P01
AR	Channel name (This is identical with the AT connection of the MSI block)	EMPFKANA
NZ	No. of cycles per data transfer	5
NL	No. of connected LI blocks	1
MEM	Diagnostics triplet	0
TGL	Diagnostics triplet	0
CVP	Connection with LI, connection CCV	>LI.CCV

NOTE In addition, the following must be configured: Central message block @MSI, message output block MSI and the message block MERF0 or a other MERFxx.

FB @MSI

I/O name	Significance	Example
CMS	Message system name	MYMELD
CMT	Message text (this is not output)	""
NOM	No. of messages which can be saved	200
SAV	Message buffer, buffered RAM	0
RP	Prefix for communication errors	0
MUN	Enable for message entries	1

FB MSI

I/O name	Significance	Example
CMS	Message system name	MYMELD
CTS	Coupling module name	D01P01
AT	Address parameter	EMPFKANA
RP	Prefix for overflow messages	0
SNV	Output, message number	1
STM	Output, message text	0
STC	Output, message text constant length	1
SSF	Output format	1
EN	Enable	1
MUN	Enable for message entries	1

FB MERF0

I/O name	Significance	Example
CMS	Message system name	MYMELD
MT	Message type	1
RP	Prefix	0
RS1	Suffix, incoming message	10001
RS2	Suffix, outgoing message	00005
EN	Message enable	1
IS1	Message trigger	16#0
SM	Save message	0

3.17.4.2 WinCC configuring software

In addition to the configuring software for the process variables, an ALARM logging configuring software must be generated. The WinCC configuring software will not be discussed here. If you require information refer to the comprehensive WinCC Configuring Manuals. We recommend the Getting Started SIMATIC WinCC Manual for an introduction into configuring WinCC. The assignment of the SIMATIC TDC message numbers to the message blocks (RS* connections) to the message

numbers, generated by WinCC, can only be identified by the "PMC message no", which is generated from the message numbers of the signal list.

3.17.5 Generating the address book using the CFC editor

To generate the signal list for WinCC, ADRIMP requires the symbol information of the SIMATIC TDC processors. For each CPU, SIMATIC TDC generates an ASCII file, which contains this information. The file name consists of the subrack names and the CPU number, separated by a "_". ".ADR" is used as extension.

The address book is generated by calling-up the required project chart, and selecting the menu items Options - Settings compilation.... Then mark the Option, Create address book, and enter OK. Call-up the menu items Chart compilation. The address book is now created when compiling. The path of the generated address book is then located via the menu items Options - Report.

NOTE All connections in SIMATIC TDC which should be controlled by WinCC have to be activated for operator control and monitoring (Refer to Manual “D7-SYS - STEP 7, CFC and SFC configuring, chapter 2.1.6.1)”.

3.17.6 Address list import tool ADRIMP

In order that WinCC can interpret the addresses of the SIMATIC TDC path names, the ADRIMP address list tool is required. The ADRIMP address list tool allows text address lists (TALI) to be listed in the WinCC data base. A precise description is provided in the User Manual TDC PMC TCP channel DLL for WinCC.

3.17.6.1 Prerequisites

A variable definition file must exist, and the SIMATIC TDC address book must have been previously generated. The variable definition file and the address book must be located in the same path. The generation path can be different, but should also be generated in this path to enhance the software transparency.

3.17.6.1.1 Generating the variable definition file

The variable definition file is a text file, which must be generated by the user. The variable definition file consists of two defined header lines (1. and 2.), followed by the assignment of symbolic names to the connection path names. The symbolic names can be freely selected, but should be the same as those used in the WinCC text fields to ensure transparency.

Excerpt from the variable definition file

E.g.: winccvar.txt

1.) [VDM:wincc]

2.) [PN:A000_1,C:\wincc\vardatei]

3.) MOTOR_EIN,ANBIND.CI.CCV

MOTOR_AUS,ANBIND.CI.YTS

3.17.6.1.2 Generating and importing a new signal list

Prerequisites

Before generating and importing the signal list, the TDC PMC TCP driver must be installed in the WinCC configuring software.

- Call-up the WinCC configuring software
- Click-on tag management
- Click-on the menu item "Add new driver"
- Select the TDC PMC TCP.chn

If no WinCC configuring software is started before ADRIMP starts, when importing the signal list, the last configuring software which was used, is used.

Execution

- Call-up ADRIMP
- Select the "File" menu item
- Select the "Probi" menu item
- Search for the variable definition file (e.g.: winvar.txt)
- Define the generation path
- Generate the signal list (e.g.: wincc.txt)
- Exit Probi

Note

ADRIMP automatically imports the signal list into the tag management of the appropriate WinCC data administration.

3.17.6.1.3 Importing an existing signal list

- Start WinCC with the required project
- Call-up ADRIMP
- Select the "File" menu item

- Select the "Open" menu item
- Select the signal list (e.g.: wincc.txt)
- Exit ADRIMP

NOTE **ADRIMP automatically imports the signal list into the tag management of the appropriate WinCC data administration.**

3.17.6.2 Checking the generated tag management in WinCC

Check the imported data, their symbolic names, data formats and path names:

- Call-up the WinCC configuring software
- Select the variables tag management
- Click-on the logical connection (corresponds to the VDM name)
- Select TDC PMC TCP
- Select the logical connection names

The logical names and path names, defined in the variables file, is displayed. The data formats are also displayed. WinCC can now access these variables.

3.17.7 Communications set-up, SIMATIC TDC-WinCC

3.17.7.1 Activating WinCC

In order to establish communications between SIMATIC TDC and WinCC, the imported data of the WinCC database must be assigned to the input/output fields of the graphic configuring software. This is realized by selecting the appropriate fields in the Graphics Designer and clicking on the interactive configuration dialog. Each field can be assigned one of the imported variables. After this assignment has been made, the File menu item is selected in the main menu, and the data are saved. Before starting runtime, the connection properties must be set.

In the Control Center

- Click-on tag management
- Click-on TDC PMC TCP
- With the righthand mouse key click-on TCP/IP Unit 1
- Click-on properties
- Click-on the properties, Channel Unit (configure PMC parameter)

- Click-on the connection

Enter the TCP/IP address for the AG (PLC) (local/remote)

Port no. (local/remote)

- Confirm with OK
- Select "File" in the Control Center
- Click-on activate

WinCC is now ready to transfer data between SIMATIC TDC and WinCC

3.17.7.2 Activating SIMATIC TDC

Power-up the configured subracks. After the subracks have run-up, the connection has been established between SIMATIC TDC and WinCC. Data is now cyclically transferred between SIMATIC TDC and WinCC.

3.18 Communications service Trace

Three different services exist for the trace value recordings:

- Simple Trace
  It only contains an output interface.
- System Trace
  It contains an interactive interface (request and reply interface). The user must develop his own tool in order to utilize the system trace. This tool must be capable of fully utilizing the system trace request/reply language. For this reason, the system trace will not be described further at this point. The interface specification and the configuring guide can be obtained from the ASI 1 R department.
- Analog Trace
  It contains no data interface.This service is only processed with the function blocks TRCC, TRCC_D und TRCC_I. A description is not be found in this configuring guide. The function is described in detail in chapter 7 of the ”Standard Function Blocks P32” catalog.

3.18.1 Simple Trace

The simple trace permits values from local processor connectors to be recorded, saved and output. The simple trace consists of a control block @TCP and one or several acquisition blocks TRP, TRP_B, TRP_D and TRP_I or TRHI. Any number of acquisition blocks can be configured in any number of sampling times.

3.18.1.1 Method of Operation of the @TCP

The @TCP consists of essentially the following tasks:

- Control of the trace value acquisition blocks TRP, TRP_B, TRP_D and TRP_I.
- Output of recorded values.
- Error Handling.

These tasks are described in detail in the following.

Control of the Trace Value Acquisition

The recording and output of trace values is controlled by the input connectors STA and TBR of the @TCP.

Recording Trace Values

A recording of the acquisition blocks is initiated in the following manner:

- R Connector: 0
- TBR Connector: 0
- STA Connector: Transition from 0 to 1

The recording is active until a transition from 1 to 0 is detected at the STA connector (a premature termination of the trace value recording by a reset will not be considered at this point). During an active recording, the @TCP is in the record mode. The recording of trace values is terminated when a transition from 1 to 0 is detected at the STA connector. If the value 0 is defined at the input connector TDC, then the trace value recording is immediately terminated. Otherwise the @TCP FB waits for the number of cycles, defined at the TDC, to terminate the trace value recording.

Output of Trace Values

An output of trace values by the @TCP can be implemented as follows after terminating a trace value recording:

- R Connector: 0
- TBR Connector: Transition from 0 to 1
- STA Connector: 0
- OUT Connector: Group telegram output (yes / no)
- CID Connector: When no group telegram output: number of connectors, whose recorded values are to be output.

During an active output, the @TCP is in the so called *output state* (a temporary status change into the wait state is possible, as long as the data interface is temporarily not accessible).

In principle, two possibilities are available for the output of trace values:

- All trace values of all acquisition blocks are transmitted together in one telegram. This is the case when the connector OUT has the value of 1.
- An acquisition block, whose trace values are to be output, is selected via the CID connector. The recorded trace values are output via the data interface in the form of one or several reply telegrams. If the recorded values, for a connector, cannot be transmitted together in one reply telegram, then further (follow up) reply telegrams are automatically transmitted. The logical sequence of the follow up telegrams is contained in two counters within the telegram: counter *number of packets*, describes the number of packets required to output all the values of the corresponding connector. The counter *packet number*, describes the number of the current packet (the numbering convention that this is based on is described in the following section). An output is terminated when either all existing trace values have been output or a transition from 1 to 0 occurs at the TBR connector.

Resetting the Simple Trace

The simple trace can be reset by setting the input connector R to 1. This action deactivates any recording or aborts any trace value outputs. The trace buffer is also reformatted, i.e. any values in the buffer are lost. Resetting remains active until the R connector is set back to 0.

Error Handling

When communication errors occur, an entry is made into the communication error panel and the @TCP is disabled (QTS connector = 0). The corresponding communication error number is additionally set at the YTS connector.

NOTE

The cause of the error as well as to a guide to resolving the error can be reviewed in the document “D7-SYS - Errorcodes for fast access”.

3.18.1.2 Method of Operation of the Acquisition Blocks

The acquisition blocks TRP, TRP_B, TRP_D and TRP_I have the following tasks:

- Recording the desired input variables (trace recording),
- Error handling.

The method of operation of all acquisition blocks is identical. They differ only with regard to the format with which the input variables are to be recorded (TRP = Real, TRP_B = BOOL, TRP_D = Double Integer, TRP_I = Integer).

Trace Recording

Each acquisition block records the values of its input connector and saves them into a ring buffer. Each acquisition block is allocated one ring buffer. The ring buffer is allocated at the start of the cyclic phase. The total trace buffer memory, whose size is determined by the connector TBL of the @TCP FB, is distributed to the ring buffers. The ring buffers are defined such that each ring buffer can record the same number of values.

The acquisition blocks record exactly one value per cycle, as long as the trace value acquisition has been activated by the central block @TCP. If connectors are recorded by acquisition blocks, configured in the interrupt layer (acyclic recording), then the acquisition time point, consisting of date and time of day, is additionally entered into the trace buffer with each trace value. No values are recorded by the acquisition blocks when the trace value acquisition is deactivated by the @TCP. If the SSF connector of the @TCP is set to 1, then the trace values are recorded by the acquisition blocks in the standardized form as floating point numbers (Real values).

Error Handling

If a communication error occurs, then an entry is made into the communication error panel and the corresponding acquisition block is disabled (QTS connector = 0). The communication error number is additionally available at the YTS connector.

NOTE The disconnection of TRP, TRP_B, TRP_D oder TRP_I after an occurrence of a communication error has no influence on the other blocks of the simple trace, as long as at least one other configured functional acquisition block is available. The cause of the error as well as the guide to resolving the error can be reviewed in “D7-SYS - Errorcodes for fast access”.

3.18.1.3 Method of Operation of the Header Block TRHI

The header block holds parameters for the group telegram. A group telegram is generated when the connector OUT of the @TCP FB has the value of 1.

The header block TRHI only affects a trace system, when a group telegram output has been configured.

The configuring of a TRHI FB has no practical consideration when no group telegram has been configured.

NOTE Configuring a TRHI FB only affects the layout of a group telegram. See also the section “Reply Telegrams”.

3.18.1.4 Simple Trace Configuring

The simple trace can be configured once (exactly 1 @TCP) or multiply (more than 1 @TCP on a P32).

Configuring Exactly <u>One</u> 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 configuring regulations are mandatory:

- @TCP and acquisition blocks of a simple trace possess the same identifiers at the TRC connector.
- The value at the TBL connector of the @TCP must be larger than zero.
- All acquisition blocks possess unique, i.e. differing, number ID's at the CID connector.
- The acquisition blocks can be configured in differing sampling times.
- At least one acquisition block (any number are permitted) must be configured.
- A maximum of 255 acquisition blocks can be configured when group telegram outputs are desired (connector OUT of the @TCP FB = 1).

<u>Example:</u> Assignment of the Initialization Connectors for Configuring Exactly one Simple Trace mit vier Erfassungsbausteinen und Sammeltelegrammausgabe.

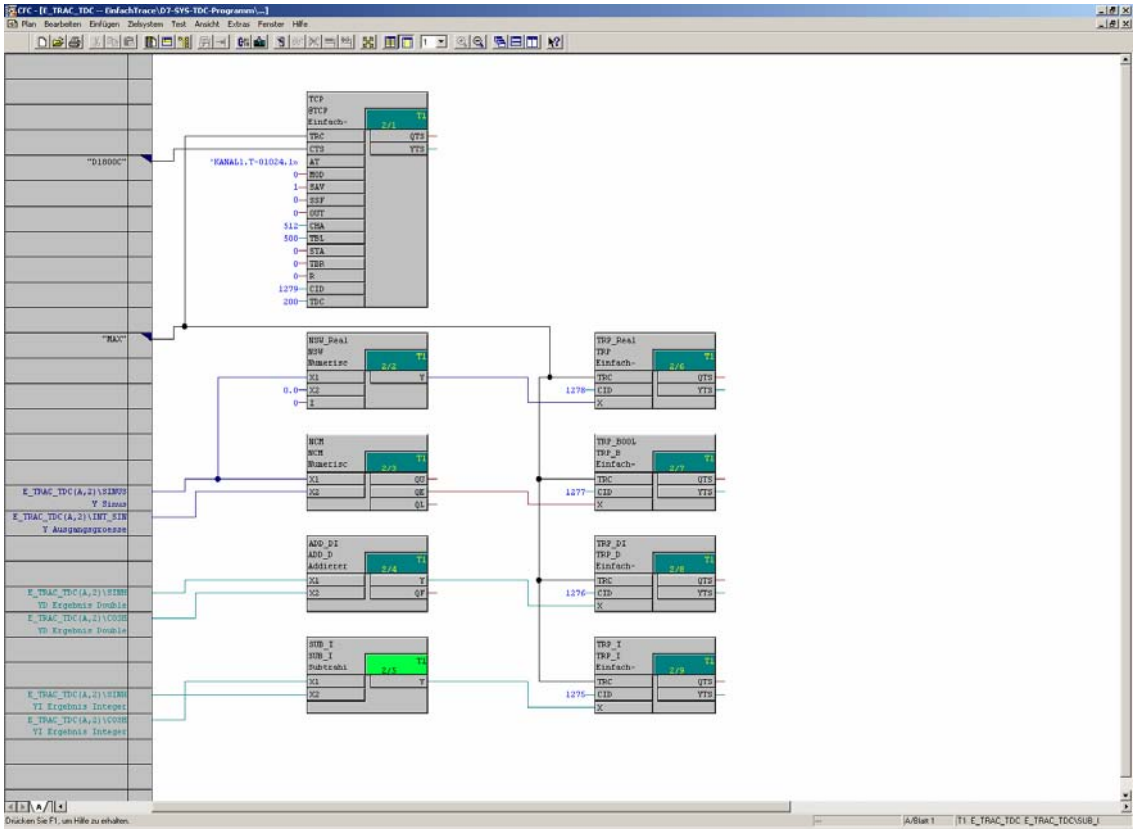


Fig. 3-94 Example for configuring of one Simple Trace

Configuring Several Simple Trace

In addition to configuring a simple trace, it is possible to configure several simple trace in parallel.

A simple trace is formed exactly by its blocks, that have identical identifiers at the TRC connectors. The following supplementary configuring regulations are valid, in addition to the configuring regulations described in the previous section.

All configured @TCP blocks must have unique, i.e. differing, TRC identifiers.

Example: Assignment of the Initialization Connectors for Configuring Two Simple Trace.

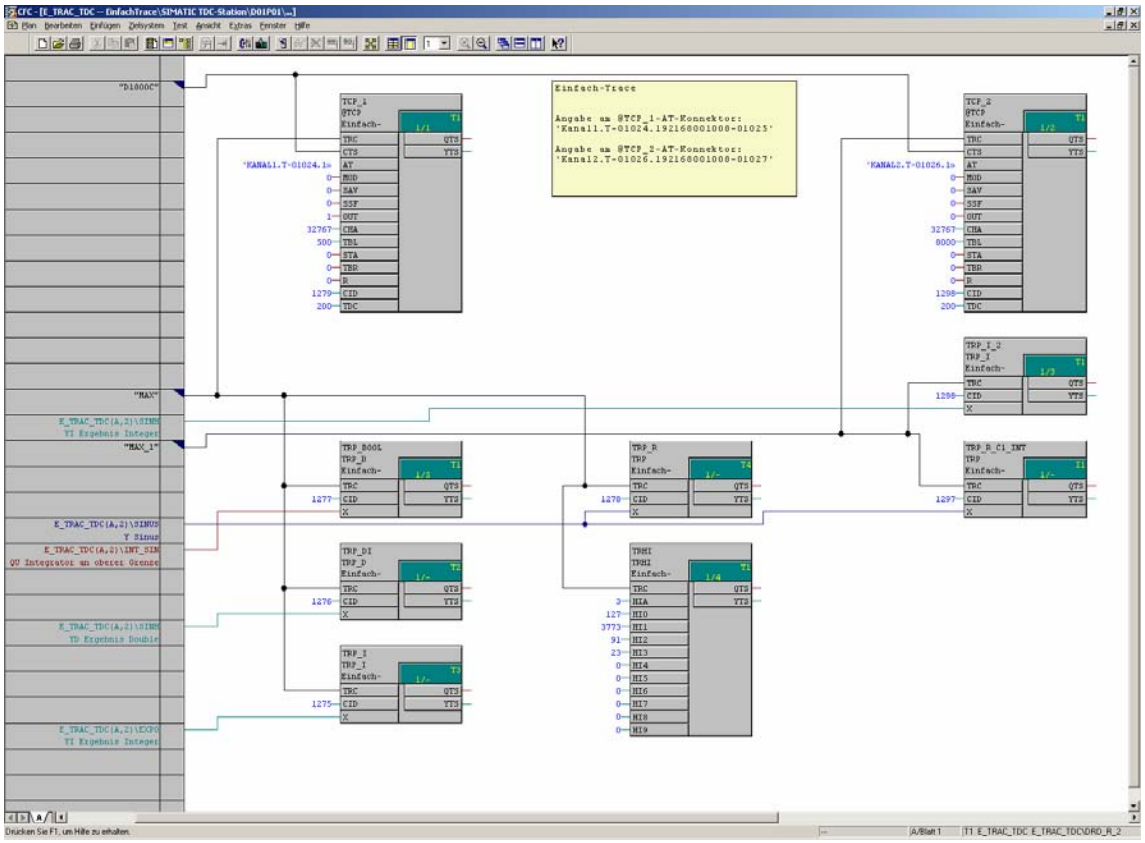


Fig. 3-95 Example for configuring of two Simple Trace

3.18.1.5 Reply Telegrams

The reply telegrams, generated by @TCP FB, are dependent upon the value of the connector OUT at the @TCP FB. If the connector has the value 0, then individual telegrams are generated and the same output format as known in the system trace is utilized. When individual telegrams are utilized, then an acquisition block is selected, whose trace values are to be output in one or several telegrams, via the CID connector of the @TCP FB. A group telegram is generated when the OUT connector has the value 1. A group telegram is characterized that it contains all the trace values of all the acquisition blocks. Information at the CID connector of the @TRP FB is then not necessary.

Individual Telegrams

An individual telegram is generated by the simple trace when the initialization connector OUT of the @TCP contains 0. The telegram has the following layout:

Byte Number	Contents	Data Format	Significance
1,2	Connector ID Number	unsigned int16	
3	Output Format	char	0=Binary, 1=Standardized
4	Connector Interpretation	char	1 - 12
5 bis 10	Last Acquisition Time Point	6 chars	
11, 12	Packet Number	unsigned int16	
13, 14	Number of Packets	unsigned int16	
15	Record Mode	char	0=Cyclic, 1=Interrupt Task
16	Connector Format	char	
17 bis 20	Sampling Time	unsigned int32	Information in1/10 ms
21, 22	Request Identifier	unsigned int16	
23, 24	No. of Blocks in Trace Buffer	unsigned int16	
25, 26	No. of Blocks in Telegram	unsigned int16	
27,28	Empty	unsigned int16	

NOTE Telegram Body: From 29 Trace Value Blocks See the following description.

The **connector ID number identifie**s the acquisition block whose trace values are to be output by the telegram. This information is identical with the information at the input connector CID of the @TCP FB.

The **output format define**s the format of the trace values in the trace value blocks. A binary format causes the connectors to output in their data types. In standardized format, floating point values of a length of 4 bytes are always utilized.

The **connector interpretation contain**s information regarding the type of recorded connector and can have the following values:

Constant	Connector Interpretation
1	N = Standard Interpretation
3	I = Whole Number
8	B = Boolean Value

The las**t acquisition time point refer**s to the last recorded trace value, as long as the recording has been implemented cyclically (this information has no significance for acyclic recordings). This trace value is the last trace value or trace value block in the telegram body with the packet number 0. The acquisition time point is transmitted in SIMADYN D format time and date (Time and Date See description „D7-SYS - SIMADYN D system and communication project planning", chapter „Kompatible user data structures".

The panels **packet number an**d **number of packets sequentiall**y number the telegram. Each reply telegram to an (implemented) output command has two counters, that contain the current number of the packet and the total number of necessary packets required for transmitting all recorded values. The numbering convention is defined such that the last packet contains the packet number 0. If, for example, three reply telegrams are required for transmitting the trace value, then the following numbering sequence occurs:

Example:

	Reply Telegram 1st.	Reply Telegram 2nd	Reply Telegram 3rd..
Packet Number	2	1	0
Number of Packets	2	2	2

The **record mode define**s whether the acquisition block is configured in a cyclic layer or an interrupt layer.

The **connector format define**s the size of the recorded trace values and can have the following values:

Constants	Significance
1	1-Byte (recorded by TRP_B block)
2	2. Byte (recorded by TRP_I block)
4	4-Byte (recorded by TRP_D block)
5	4-Byte standardized (recorded by TRP block)

The **sampling time define**s the sampling time of the corresponding acquisition block in 1/10 ms.

The **request identifier contain**s information regarding the layer in which the corresponding acquisition blocks were configured:

Sampling	**Time Constants**
T1	0
T2	1
T3	2
T4	3
T5	4

Interrupt Layer Constants

I1	5
I2	6
I3	7
I4	8
I5	9

The **number of trace value blocks in the trace buffer define**s how many recorded values are contained in the trace buffer for this acquisition.

The **number of trace value blocks in the telegram define**s how many trace value blocks are contained in the current telegram.

The **trace value blocks contai**n the data of a connector, recorded up to a certain time point (possibly expanded with dummy bytes). The structure of a trace value block is dependent upon the connector format and the recording mode of the acquisition block (recording this connector) and the configured output format of the @TCP. All the possible trace value blocks are described in the following table (all number definitions in bytes).

Recording Mode	Output Format	Connector Format	Connector Value	Dummy Bytes	Time Point of Recording	Dummy Bytes	Block Length
Cyclic	0	1	1	-	-	-	1
Cyclic	0	2	2	-	-	-	2
Cyclic	0	4	4	-	-	-	4
Cyclic	0	5	4	-	-	-	4
Acyclic *	0	1	1	1	6	-	8
Acyclic *	0	2	2	-	6	-	8
Acyclic *	0	4	4	-	6	2	12
Acyclic *	0	5	4	-	6	2	12
Cyclic	0	1,2,4,5	4	-	-	-	4
Acyclic *	0	1,2,4,5	4	-	6	2	12

A trace value block consists of the connector value, a dummy byte (optional), the time point of the acquisition (only for acyclic recording, output in SIMADYN D format *time and date*) and further dummy bytes (optional). The output format is defined by the SSF connector of the @TCP block. The connector format is defined by the acquisition block type.

* The connector value is recorded by an acquisition block in an interrupt layer (I1--I5).

Example:

Structuring the values in the telegram body for a 1 byte connector, recorded by an acquisition block acyclically (interrupt layer). The output is implemented in standardized form as a floating point number.

Byte	Significance
29 ... 32	Trace Value to Time Point t1
33 ... 38	Acquisition Time Point t1
39, 40	Dummies
41 ... 44	Trace Value to Time Point t2
45 ... 50	Acquisition Time Point t2
51, 52	Dummies
...	...

Group Telegrams

A group telegram is generated by @TCP FB, when the value 1 is present at the initialization connector OUT of the @TCP FB.

A group telegram basically consists of **one telegram**. This signifies that the output channel must be correspondingly dimensioned to accept all trace values at one time (CHA connector of the @TCP FB). If the channel is too small, then the @TCP FB reverts to the status OFF at the start of the cyclic phase. A communication error message is generated, whereby the error message contains the minimum channel size information. The minimum size of the output channel can be calculated according to the following formula:

CHA >= 12 +4 * HIA +12 * TRPx +4 * TBL

Index:

- CHA -Minimum channel size in bytes (information at the connector CHA of the @TCP)
- HIA -Information at the connector HIA of the TRHI FB (0 when TRHI FB is not configured)
- TRPx -Number of configured (and correctly initialized) acquisition blocks
- TBL -Size of the trace buffer (information at the connector TBL of the @TCP)

The group telegram layout is made up as follows:

Telegram Header:

Byte Number	Content	Data Format	Significance
1	Number of Active TRPx char	char	
2	Output Format	char	See individual telegram
3 to 8	Last Acquisition Time Point	6 chars	
9 to 12	Parameter 0	unsigned int32	First parameter TRHI--FB
. . .			
9 + (n-1) * 4	Parameter n-1	unsigned int32	n--th. parameter TRHI--FB

Telegram Body:

Byte-Nr.	Inhalt	Datenformat	Bedeutung
9 + n * 4	Connector ID Number	unsigned int16	See individual telegram
11 + n * 4	Connector Interpretation	char	See individual telegram
12 + n * 4	Connector Format	char	See individual telegram
13 + n * 4	Sampling Time	unsigned int32	See individual telegram
17 + n * 4	Request Identifier	unsigned int16	See individual telegram
19 + n * 4	Number of Trace Value Blocks	unsigned int16	See individual telegram
21 + n * 4	Number of Trace Value Blocks		See individual telegram

A telegram body exists for each acquisition block. The telegram bodies are sorted according to the log on sequence of the corresponding acquisition blocks.

The type of data within the group telegram is basically identical to the type of data in the individual telegram.

The **number of active TRPx corresponds** to the number of telegram bodies in the group telegram. The allocation of the telegram bodies to the acquisition blocks is then possible via the connector ID number, which corresponds to the CID information of the acquisition blocks.

The **last acquisition time point i**n the telegram header of a group telegram receives the time point of detection of the stop signal. The last acquisition time point in the individual telegrams corresponds to the time point of the recording of the last trace value in the trace buffer and can therefore differ in acquisition blocks.

The **parameter 1- n o**f the TRHI FB is only entered when it has also been configured. The connector HIA of the TRHI FB indicates how many parameters are inserted into the header.

Warning: no information exists in the group telegram of whether and how many parameters have been transferred from the TRHI FB. The interpretation of the group telegram is therefore dependent upon the configuring. The parameters contain supplementary information, of which the number and its interpretation is left up to the user.

Each telegram body starts with an offset, with regard to the start of the telegram, which is divisible by 4. Therefore telegram bodies are filled with dummy bytes under certain circumstances.

Example: a group telegram is generated in which three trace values of a TRP_B FB exists. The TRP_B FB are operated in cyclic recording mode and with a binary output format, such that the corresponding trace value block has a size of 1 byte. Therefore the panel *number of trace value blocks,* in the telegram body, has the value of 3 and is followed by 3 bytes of trace value blocks, which are existent in the telegram. A dummy byte is now inserted. The next telegram body starts behind this dummy byte. No dummy bytes are inserted behind the last telegram body.

Index

$

$ signals 2-17

7

7-segment display 2-45
  Acknowledge error 2-45

A

Application example PROFIBUS DP 3-82
  Configuring slaves 3-98
Assigning a name 2-3

B

Basic clock cycle 2-25
Basic CPU clock cycle 2-19
Basic information, communications 3-2
Behavior under fault conditions 2-39

C

CFC chart (Continuous Function Chart) 2-7
CFC editor 2-7
  Margins 2-11
  Parameterizing dialogs 2-8
COM PROFIBUS 3-77
Communicaions
  SIMATIC Operator Panels 3-176
Communication blocks
  Address connections AT, AR, US 3-7
  Central coupling blocks 3-15
  Firmware status, ECL, ECO connection 3-12
  Initialization input CTS 3-7
  MOD connection 3-9
  Status display, output YTS 3-13
  Transmitter and receiver 3-15
Communication utilities
  Overview 3-6
Communications
  WinCC via SINEC H1 3-216
Communications buffer coupling 3-21
Configuring SIMADYN D stations 2-4
Consistency check 2-7
Coupling modules

Number in the subracks 3-17
Couplings
Data interface 3-18
Mode of operation 3-13
Net data structures 3-16
Overview 3-2
Couplings on the subrack 3-19
CPU synchronization 2-25
Configuring the CPU basic clock cycle 2-26
Configuring the interrupt task 2-28
Cycle errors 2-31
Eliminating 2-32

D

Data consistency 2-21
Data transfer mode
Handshake 3-9
Image 3-12
Multiple 3-11
Overview 3-8
Refresh 3-9
Select 3-10
Deadtimes 2-19
Direct CPU-CPU coupling 3-20
Download in the RUN status 2-15

E

Error differentiation 2-36

F

Fast $ signal 2-18
Features
Computation times of the operating system 2-33
Cyclic tasks 2-33
Interrupt tasks 2-33
Memory requirement of the operating system 2-34
Function block
Assignment to interrupt tasks 2-9
Comments 2-12
Function blocks
Assigning to cyclic tasks 2-9

H

hardware address 2-12
Hardware timer 2-26
HWConfig 2-5
Parameterizing dialogs 2-5

I

Initialization 2-35

Interconnecting 2-12
Interrupt-controlled processing 2-36

L

Libraries 2-4
Limited number of interconnections 2-20
Loading the user program
  Offline loading 2-13
  Online loading 2-13
Local CPU coupling 3-19

M

Message system 3-138
  Communications error message 3-145
  Entry logic 3-138
  Error or alarm message 3-143
  Message entry blocks 3-138
  Message format 3-143
  Message formats 3-146
  Message type description 3-143
  Messages 3-140, 3-143
  Output format 3-151
  Overflow message 3-145
  System error 3-146
  System error message 3-145
MPI coupling 3-101
  Configuring 3-101

O

Operating system components 2-34
Operator Panels (SIMATIC) 3-176

P

Pointer-based communications blocks
  Applications 3-162
  Associated function blocks 3-163
  Configuring information and instructions 3-164
  Examples 3-166
  Features 3-162
  Pointer interface 3-164
  Principal mode of operation 3-162
Pointer-based communications blocks
  Introduction 3-161
Process data 3-153
  Blocks CRV, CTV 3-157
  Channel marshalling blocks 3-157
  Channels 3-160
  Configuring example 3-155
  Diagnostics 3-159
  Distribution block 3-158
  Function blocks 3-153

Virtual connections 3-153
Process image 2-21
Implementation 2-22
PROFIBUS DP 3-65
Address connection 3-54, 3-67
Application example 3-91
COM database 3-91
COM PROFIBUS 3-91
Communications module SS52 3-91
Configuring 3-54, 3-66, 3-88
Configuring CFC 3-85
Diagnostic data 3-73
Download COM database 3-99
Error class 3-64, 3-81
Hardware and software 3-85
LED 3-80
Memory SS52 3-98
Parameterization 3-95
Parameterizing 3-91
SIEMENS DP slaves 3-62, 3-75
SYNC/FREEZE 3-56, 3-68
Transmit- and receive blocks 3-87
Typical configuration 3-83
Pseudo comments 2-12

S

Service 3-171
Function block SER 3-172
System load 3-173
Service utility
SER function block 2-38
System loading, response times 2-38
Signal transfer 2-15
Task 2-16
SIMATIC Operator Panel
Alarm message 3-180
Block I/O 3-178
Computation times, function blocks 3-183
Configuration HWConfig 3-177
Configuring CFC 3-177
Event 3-179
Example of a configuration 3-176
Function keyboard 3-180
Initialization 3-178
Interface area 3-181
ProTool/Lite Configuring 3-182
Requirements 3-176
Symbol table 3-181
Slot number 2-6
Standard mode 2-23
Subrack coupling CP52M0
Applications 3-22
Behavior when powering-up and powering-down 3-23
Configuring 3-24

Performance data 3-27
Subrack coupling CP53M0
Configuring 3-34
General 3-28
Hardware structure 3-31
Initialization and monitoring 3-28
Response when powering-up the master subrack 3-32
Response when shutting down a coupling partner 3-32
Restart frequency 3-33
Restrictions 3-35
Scope of supply 3-31
Symbol table 3-181
Synchronization configuration 2-29
System chart 2-7
System mode 2-22
System status user stop 2-15

T

Table function 3-102
Introduction 3-102
Task administrator 2-19, 2-31
Task processing 2-19, 2-21
Time of day synchronization 3-174
Troubleshooting
Background processing 2-43

U

Utility programs 2-37

V

Virtual connections 3-153

W

WinCC via SINEC H1 3-216