Concept 2.6

User Manual

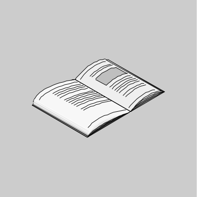
12/2010

33002204.12

www.schneider-electric.com

Schneider Electric

Table of Contents



Safety Information 19

About the Book. 21

Chapter 1 General description of Concept 23

1.1 General description of Concept. 24

Introduction 25

PLC hardware configuration 27

PLC Hardware Package Contents in Concept S, M and XL 28

1.2 Programming. 29

General information. 30

Libraries. 31

Editors 33

Online functions. 37

Communication 38

Secure Application 39

Utility program 41

Chapter 2 New Performance Attributes of Concept 2.6 in Comparison with Concept 2.5 45

New Performance Attributes of Concept 2.6 Compared with Concept 2.5 46

New performance attributes of Concept 2.6 SR2 in comparison with Concept 2.6 SR1 52

New performance attributes of Concept 2.6 SR3 in comparison with Concept 2.6 SR2. 55

Chapter 3 Project structure. 57

Project Structure and Processing 58

Programs. 64

Sections. 68

Configuration data 73

Chapter 4 Creating a Project. 75

Overview 76

Step 1: Launching Concept. 77

Step 2: Configuring the PLC 78

Step 2.1: Required Configuration 79

Step 2.2: Optional Configuration 81

Step 3: Creating the User Program 85
Step 4: Save 88
Step 5: Perform Memory Prediction. 89
Step 6: Loading and Testing 90
Step 7: Optimize and Separate 95
Step 8: Documentation 97
Chapter 5 PLC configuration 99
5.1 General information about hardware configuration 100
General information 101
Proceed in the following way with the configuration 102
5.2 Configuration in OFFLINE and ONLINE mode 103
General information 104
Available Functions in OFFLINE and ONLINE Modes. 105
5.3 Unconditional Configuration. 107
Precondition. 108
PLC selection. 109
CPU Selection for the PLC Type 110
PLC memory mapping 113
Loadables 114
Segment manager 117
I/O Map 119
5.4 Optional configuration 122
Settings for ASCII Messages. 123
Making Additional Functions Available in the Configurator 124
Data Exchange between Nodes on the Modbus Plus Network 125
How many words are really used when data is received (Peer Cop) . . . 126
Protecting Data in the State RAM before Access 128
Parameterize interfaces. 129
Special Options 131
5.5 Backplane Expander Config 133
Generals to Backplane Expander 134
Edit I/O Map. 135
Error handling 136
5.6 Configuration of various network systems 137
Configure INTERBUS system 138
Configure Profibus DP System 139
Configure Ethernet. 141
RTU extension 143
Ethernet I/O Scanner 144
How to use the Ethernet / I/O Scanner 149
5.7 Quantum Security Settings in the Configurator 151
Quantum Security Parameters. 151

Chapter 6 Main structure of PLC Memory and optimization of memory 155
6.1 Main structure of the PLC Memory 156
General structure of the PLC Memory. 156
6.2 General Information on Memory Optimization. 157
Possibilities for Memory Optimization 158
PLC-Independent 159
6.3 Memory Optimization for Quantum CPU X13 0X and 424 02 162
General Information on Memory Optimization for Quantum CPU X13 0X and 424 02 163
Selecting Optimal EXEC File. 165
Using the Extended Memory (State RAM for 6x references) 169
Harmonizing the IEC Zone and LL984 Zone. 171
Harmonizing the Zones for Global Data and IEC Program Memory 173
6.4 Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B). . . . 176
General Information on Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B) 177
Harmonizing IEC Zone and LL984 Zone. 179
Harmonizing the Zones for Global Data and IEC Program Memory (CPU 434 12(A) / 534 14 (A/B)) 184
6.5 Memory optimization for Compact CPUs 187
General Information on Memory Optimization for Compact CPUs 188
Harmonizing IEC Zone and LL984 Zone. 190
Harmonizing the Zones for Global Data and IEC Program Memory (Compact) 194
6.6 Memory optimization for Momentum CPUs. 197
General Information on Memory Optimization for Momentum CPUs 198
Selecting Optimal EXEC file 200
Harmonizing the Zones for Global Data and IEC Program Memory (Momentum) 201
6.7 Memory optimization for Atrium CPUs 204
General Information on Memory Optimization for Atrium CPUs 205
Use of IEC. 207
Harmonizing the Zones for Global Data and IEC Program Memory (Atrium) 209
Chapter 7 Function Block language FBD. 213
7.1 General information about FBD Function Block 214
General information on Function Block language FBD 214
7.2 FBD Function Block objects 215
Functions and Function Blocks (FFBs) 216
Link 220
Actual parameters 221
Text Object 223

7.3 Working with the FBD Function Block langauge 224
Positioning Functions and Function Blocks 225
FFB Execution Order 227
Configuring Loops 230
7.4 Code generation with the FBD Function Block language 232
Code Generation Options 232
7.5 Online functions of the FBD Function Block language. 233
Online Functions 233
7.6 Creating a program with the FBD Function Block language 235
Creating a Program in the FBD Function Block Language 235
Chapter 8 Ladder Diagram LD 239
8.1 General information about Ladder Diagram LD. 240
General Information about the LD Ladder Diagram Language 240
8.2 Objects in Ladder Diagram LD. 242
Contacts. 243
Coils. 245
Functions and Function Blocks (FFBs) 248
Link 253
Actual Parameters 254
Text object 256
8.3 Working with the LD Ladder Diagram 257
Positioning Coils, Contacts, Functions and Function Blocks 258
Execution sequence 260
Configuring Loops 262
8.4 Code generation with LD Ladder Diagram. 264
Code Generation Options 264
8.5 Online functions with the LD Ladder Diagram 265
Online Functions 265
8.6 Creating a program withLD Ladder Diagram 267
Creating a Program in LD 267
Chapter 9 Sequence language SFC 271
9.1 General information about SFC sequence language. 272
General information about SFC language 272
9.2 SFC sequence language elements 274
Step 275
Action. 278
Transition 280
Transition section. 282
Link 284
Jump 285
Alternative Branch 287
Alternative connection 289
Parallel branch. 290
Parallel connection. 291
Text object 292

9.3 Working with the SFC Sequence Language 293
General information on editing objects 294
Declaring step properties 297
Declaring actions. 299
Identifier. 302
Declaring a Transition 305
Alias Designations for Steps and Transitions 307
9.4 Online functions of the SFC sequence language 310
Animation 311
Controlling a Step String 313
Learn monitoring times 316
Transition diagnosis 319
Chapter 10 Instruction list IL. 321
10.1 General information about the IL instruction list 322
General Information about the IL Instruction List 322
10.2 Instructions 324
General information about instructions 325
Operands. 327
Modifier 329
Operators 331
Tag 334
Declaration (VAR...END_VAR) 336
Comment. 338
10.3 IL instruction list operators 339
Load (LD and LDN) 340
Store (ST and STN). 341
Set (S) 342
Reset (R) 344
Boolean AND (AND, AND (), ANDN, ANDN ()). 346
Boolean OR (OR, OR (), ORN, ORN ()) 348
Boolean exclusive OR (XOR, XOR (), XORN, XORN ()) 350
Invert (NOT) 352
Addition (ADD and ADD ()) 353
Subtraction (SUB and SUB ()). 354
Multiplication (MUL and MUL()). 355
Division (DIV and DIV ()) 357
Compare on "Greater Than" (GT and GT ()). 359
Compare to "Greater than/Equal to" (GE and GE ()) 360
Compare to "EQual to"(EQ and EQ ()) 361
Compare to "Not Equal to" (NE and NE ()) 362
Compare to "Less than/Equal to" (LE and LE ()). 363
Compare to "Less Than"(LT and LT ()) 364

Jump to label (JMP, JMPC and JMPCN) 365
Call Function Block/DFB (CAL, CALC and CALCN) 368
FUNCNAME 369
Right parenthesis ")" 370
10.4 Call up of functions, Function Blocks (EFBs) and Derived Function Blocks (DFBs) 371
Use of Function Blocks and DFBs 372
Invoking a Function Block/DFB 374
Function call 379
10.5 Syntax check and Code generation 381
Syntax Check 382
Code generation 384
10.6 Online functions of the IL instruction list 385
Animation 386
Monitoring field 389
10.7 Creating a program with the IL instruction list 390
Creating a program in the IL instruction list 390
Chapter 11 Structured text ST 393
11.1 General information about structured Text ST 394
General Information about the ST Structured Text 394
11.2 Expressions 396
Operands 397
Operators 399
11.3 Operators of the programming language of structured ST text 402
Use of parentheses "()" 403
FUNCNAME 404
Exponentiation (**) 405
Negation (-) 406
Complement formation (NOT) 407
Multiplication (*) 408
Division (/) 409
Modulo (MOD) 410
Addition (+) 411
Subtraction (-) 412
Comparison on "Greater Than" (>) 413
Comparison on "Greater than/Equal to" (>=) 414
Comparison with "EQual to" (=) 415
Comparison with "Not Equal to" (<>) 416
Comparison with "Less Than"(<) 417
Comparison with "Less than or Equal to" (<=) 418
Boolean AND (AND or &) 419
Boolean OR (OR) 420
Boolean Exclusive OR (XOR) 421

11.4 Assign instructions 422
Instructions 423
Assignment 424
Declaration (VAR...END_VAR) 426
IF...THEN...END_IF. 428
ELSE 429
ELSIF...THEN 430
CASE...OF...END_CASE 431
FOR...TO...BY...DO...END_FOR 432
WHILE...DO...END_WHILE. 435
REPEAT...UNTIL...END_REPEAT 437
EXIT 438
Empty instruction. 439
Comment. 440
11.5 Call up of functions, Function Blocks (EFBs) and Derived Function Blocks (DFBs) 441
Function Block/DFB Invocation 442
Function Invocation 446
11.6 Syntax check and code generation 448
Syntax Check 449
Code generation 450
11.7 Online functions of the ST programming language 451
Online functions. 451
11.8 Creating a program with the structured ST text. 452
Creating a program in structured ST text 452
Chapter 12 Ladder Logic 984 455
12.1 General about Ladder Logic 984. 456
General about Ladder Logic 984. 456
12.2 Working with Ladder Logic 984 458
Entering and Editing Logic Objects 459
Entering and Editing Variables 461
Ladder and Network Editing 464
Reference Zoom and DX Zoom 466
Search and Replace 468
12.3 Subroutines 469
Subroutines 469
12.4 Equation Network Editor 471
Introduction 472
Equation Editing 474
Syntax and Semantics. 476
12.5 LL984 Programming Modes 480
LL984 Programming Modes 480

Chapter 13 DFBs (Derived Function Blocks) 481
13.1 DFBs (Derived Function Blocks) 482
General information about DFBs (Derived Function Blocks) 483
Global / Local DFBs 485
Use of variables in DFBs 487
Combined Input/Output Variables (VARINOUT Variables) 489
Global Variables. 496
Creating Context Sensitive Help (Online Help) for DFBs. 499
13.2 Programming and calling up a DFB 501
At a Glance 502
Creating the DFB 503
Creating the Logic in FBD Function Block Language 504
Creating the Logic in LD Ladder Diagram 507
Creating the Logic in IL Instruction List 511
Creating the Logic in ST Structured Text. 513
Calling up a DFB in the FBD Function Block dialog. 515
Calling up a DFB in Ladder Diagram LD 517
Calling up a DFB in the IL instruction list 519
Calling up a DFB in structured text ST. 520
Chapter 14 Macros 521
14.1 Macro 522
Macros: general 523
Global / Local Macros 525
Exchange marking 527
Creating Context Sensitive Help (Online Help) for Macros 529
14.2 Programming and calling up a macro 531
At a Glance 532
Occupying the macro 533
Creating the logic 534
Calling up a macro from an SFC section 537
Calling a macro from an FBD/LD section. 540
Chapter 15 Variables editor 543
General 544
Declare variables 545
Searching and replacing variable names and addresses 548
Searching and Pasting Variable Names and Addresses 552
Exporting located variables 556
Chapter 16 Project Browser 557
General information about the Project Browser 558
Detailed view in the project browser 561
Operating the Project Browser. 563

Chapter 17 Derived data types 565
17.1 General information on Derived Data Types 566
Derived Data Types. 567
Global / Local Derived Data Types 570
Extended Data Type Definition (larger than 64 Kbytes) 572
17.2 Syntax of the data type editor 574
Elements of the Derived Data Types. 575
Key Words. 577
Names of the derived datatypes 582
Separators. 583
Comments. 585
17.3 Derived data types using memory. 586
Use of Memory by Derived Data Types 586
17.4 Calling derived data types. 588
Calling Derived Data Types. 588

Chapter 18 Reference data editor. 595
General Information about the Reference Data Editor 596
Converting RDE templates 598
Changing signal states of a Located variable 600
Cyclical Setting of Variables 602
Unconditional locking of a section. 605
Animation 606
Replacing variable names. 608
Load reference data 609

Chapter 19 ASCII Message Editor. 611
19.1 ASCII Editor Dialog 612
Generals to ASCII editor dialog. 613
Text 614
Variables 615
Control code 616
Spaces 617
Carriage Return. 618
Flush (buffer). 619
Repeat. 620
19.2 User Interface of ASCII Message Editor 621
How to Use the ASCII Message Editor 622
Message Number 624
Message Text 626
Simulation Text 627
19.3 How to Continue after Getting a Warning 628
How to Continue after Getting a Warning 628
19.4 ASCII Editor in Offline/Combination/Direct Modes 629
ASCII Message Editor in Offline/Combination/Direct Modes 629

Chapter 20 Online functions. 631
20.1 General information about online functions 632
General information 632
20.2 Connect to PLC 633
General 634
Presettings for ONLINE operation 637
Modbus Network Link 638
Modbus Plus Network 639
Modbus Plus Bridge. 644
TCP/IP-Network Link 646
Connecting IEC Simulator (32 bit) 647
State of the PLC 648
20.3 Setting up and controlling the PLC 649
General Information 650
Setting the Time for Constant Scans 651
Single Sweeps 652
Deleting memory zones from the PLC. 654
Speed optimized LL984-Processing 655
Save To Flash 656
Reactivate flash save. 659
Set PLC Password. 660
20.4 Selecting Process information (status and memory) 663
General information 664
PLC state 665
Memory Statistics. 667
20.5 Loading a project 670
General information 671
Loading 672
Download Changes 674
Uploading the PLC. 677
Upload Procedure 679
20.6 Section animation 681
IEC-Sections animation 682
LL984 Programming Modes. 684
20.7 Online Diagnosis 685
Diagnostics Viewer 685
20.8 Logging Write Access to the PLC 687
Logging and Encrypted Logging 687
Chapter 21 Import/Export 691
21.1 General Information about Import/Export. 692
General Information about Import/Export. 692
21.2 Exporting sections 694
Exporting Sections. 694
21.3 Exporting variables and derived data types. 697
Exporting variables and Derived Data Types. 697

21.4 Section import 698
Importing Sections. 699
Procedure for importing sections. 703
Importing IL and ST Programs to FBD, SFC, IL or ST Sections (with Conversion) 709
Importing (insert file) IL and ST programs into IL or ST sections 712
Procedure for "Copying" an IL section from an existing project into a new project.. 713
Procedure for converting FBD sections from an existing project into IL sections of a new project. 714
21.5 Variables import 716
Importing Variables in "Text Delimited" Format. 717
Importing structured variables 720
Importing variables in Factory Link format 724
Multiple Address Assignment after Variable Import 725
21.6 Import/Export of PLC Configuration 726
Import/Export of PLC Configuration using Concept 727
Import/Export of PLC Configuration using Concept Converter 728
Chapter 22 Documentation and Archiving. 729
22.1 Documentation of projects, DFBs and macros 730
Documentation contents 731
Documentation Layout 732
Defining Page Breaks for Sections 735
Use of keywords 739
22.2 Managing projects, DFBs and macros 740
Archiving projects, used DFBs, EFBs and data type files 741
Deleting projects, DFBs and macros. 743
Chapter 23 Simulating a PLC 745
23.1 Simulating a PLC (16-bit simulator). 746
Simulating a Controller 746
23.2 Simulating a PLC (32-bit simulator). 748
Concept-PLCSIM32 749
Simulating a PLC. 751
Simulating a TCP/IP interface card in Windows 98. 753
Simulating a TCP/IP interface card in Windows NT 754
Chapter 24 Concept Security 757
General Description of Concept Security 758
Access Rights 760
Changing Passwords 767
Activating Access Rights. 769
Protecting Projects/DFBs 770
Appendices 773

Appendix A Tables of PLC-dependent Performance Attributes. 775
Performance of Quantum. 776
Performance Attributes of Compact. 781
Performance Attributes of Momentum 786
Performance Attributes of Atrium. 792
Appendix B Windows interface 799
B.1 Window 800
Window Types 801
Elements of a window 803
B.2 Menu commands 806
Menu commands 806
B.3 Dialog boxes 808
Dialog boxes 808
B.4 Generating a project symbol 811
Creating a Project Symbol in a Program Group. 811
B.5 Online help. 813
At a Glance 814
How the Online Help is set out. 815
Appendix C List of symbols and short cut keys 819
C.1 Icon bar 820
General icon bar 821
Icon bar in the FBD editor 822
Icon bar in the SFC-Editor 823
Icon bar in the LD editor. 825
List of Symbols in the IL and ST Editor 827
List of Symbols in the LL984-Editor 828
Icons in PLC Configuration 829
Toolbar in the RDE Editor 830
Toolbar in the Project Browser. 831
C.2 Short cut keys 832
General Short Cut Keys 833
Short Cut Keys in the IL, ST and Data Type Editor 834
Short Cut Keys in the FBD and SFC Editor 837
Shortcut keys in the LD-Editor 841
Short Cut Keys in the LL984-Editor 847
Appendix D IEC conformity 849
D.1 What is the IEC 1131-3 standard?. 850
General information about IEC conformity. 851
IEC Conformity Test. 852
D.2 IEC standards tables 853
Common elements. 854
IL (AWL) language elements 861
ST language elements. 863

Common graphic elements 865
LD (KOP) language elements 866
Implementation-dependent parameters 868
Error causes 871
D.3 Expansions of IEC 1131-3. 873
Expansions of IEC 1131-3. 873
D.4 Text language syntax 875
Text Language Syntax 875
Appendix E Configuration examples. 877
E.1 Quantum Example - Remote Control with RIO 878
Editing local drop. 879
Editing Remote Drop 884
E.2 Quantum Example - Remote control with RIO (series 800) 887
Editing Local Drop. 888
Editing Remote Drop 893
Editing Remote Drop 897
E.3 Quantum Example - Remote Control with DIO 900
Editing Local Drop. 901
Editing Local Drop. 906
E.4 Quantum Example – INTERBUS Control 910
General Information. 911
Editing Local Drop. 912
E.5 Quantum Example - SY/MAX Controller 916
Editing Local Drop. 917
Editing Remote Drop 922
E.6 Quantum Example - Profibus DP Controller 925
General Information. 926
Profibus DP Export Settings in SyCon 927
Editing Local Drop. 929
Importing Profibus DP Configuration. 934
E.7 Quantum-Example - Peer Cop 941
Generals to Peer Cop 942
Configuration of Peer Cop. 944
Global data transfer. 946
Specific data transfer. 948
E.8 Compact Example 950
Editing Local Drop. 950
E.9 Atrium Example – INTERBUS Controller 955
General 956
INTERBUS export settings in CMD. 957
Edit local I/O drop 958
Edit remote I/O drop (import INTERBUS configuration) 962

E.10 Momentum Example - Remote I/O Bus 965
General Information 966
Editing local drop 967
Example 10 – Editing Remote Drops (I/O Bus) 971
E.11 Momentum Example - Ethernet Bus System 974
Configure Ethernet 975
Network Configuration in Different Operating Systems 976
Editing local drop 986
Create online connection 990

Appendix F Convert Projects/DFBs/Macros 991
Converting projects/DFBs 991

Appendix G Concept ModConnect 995
G.1 Introduction 996
Introduction 996
G.2 Integration of Third Party Modules 997
Integrating new Modules 998
Removing Modules 999
G.3 Use of third party module in Concept 1000
Use of Third Party Modules in Concept 1000

Appendix H Convertion of Modsoft Programs 1001
Introduction 1002
How to Convert a Modsoft Program 1004
Exceptions 1005

Appendix I Modsoft and 984 References 1007
Modsoft Keys with Concept Equivalents 1008
Modsoft Function Compatibility 1010

Appendix J Presettings when using Modbus Plus for startup 1011
Installing the SA85/PCI85 with Windows 98/2000/XP 1012
Installing the SA85/PC185 in Windows NT 1016
Installing the Modbus Plus Driver in Windows 98/2000/NT 1018
Virtual MBX Driver for 16 bit application capability with Windows 98/2000/NT 1019
MBX Driver for connection between ModConnect Host interface adapters and 32 bit applications with Windows 98/2000/NT 1020
Remote MBX - Driver for Remote Operation 1021
Ethernet MBX - Driver for Modbus Plus Function via TCP/IP 1022
Establishing the hardware connection. 1024

Appendix K Presettings when using Modbus for startup 1025
Interface Settings in Windows 98/2000/XP 1026
Interface Settings in Windows NT 1028
Setting up the hardware connection 1029
Transfer problems 1030

Appendix L Startup when using Modbus with the EXECLoader . . . 1031
Quantum first startup with EXECLoader 1032
Compact first startup with EXECLoader 1036
Momentum first startup for IEC with EXECLoader 1040
Momentum first startup for LL984 with EXECLoader 1044
Appendix M Startup when using Modbus with DOS Loader 1049
Quantum first startup with DOS Loader 1050
Compact first startup with DOS Loader. 1053
Momentum first startup for IEC with DOS Loader. 1056
Momentum first startup for LL984 with DOS Loader. 1059
Appendix N Startup when using Modbus Plus with the EXECLoader 1063
Quantum first startup with EXECLoader 1064
Compact first startup with EXECLoader 1068
Atrium first startup with EXECLoader 1072
Momentum first startup for IEC with EXECLoader 1076
Momentum first startup for LL984 with EXECLoader 1080
Appendix O Startup when using Modbus Plus with DOS Loader . . . 1085
Quantum first startup with DOS Loader 1086
Compact first startup with DOS Loader. 1089
Atrium first startup with DOS Loader. 1092
Momentum first startup for IEC with DOS Loader. 1095
Momentum first startup for LL984 with DOS Loader. 1098
Appendix P EXEC files 1101
Loading Firmware 1101
Appendix Q INI Files 1105
Q.1 Settings in the CONCEPT.INI File. 1106
General information on the Concept INI file 1107
INI Print Settings 1108
INI Settings for Register Address Format, Variable Storage and Project Name Definition. 1109
INI-settings for path information and global DFBs [Path] [Upload] 1110
Representation of Internal Data in the INI File 1113
INI Settings for the LD Section 1114
INI Settings for Online Processing [Colors]. 1115
INI Settings for Warning Messages and the Address Format 1116
INI Security Settings 1117
INI-Settings for the RDE behavior. 1118
INI settings for the Options> Toolsmenu. 1119
Q.2 Settings in the Projectname.INI File 1120
General Information for Projectname.INI File 1121
INI Settings for the Event Viewer [Online Events]. 1122
INI-Settings for the Online-Backup [Backup]. 1123

Appendix R Interrupt Processing 1125

R.1 General information about interrupt sections. 1126

General Information about Interrupt Processing 1126

R.2 Interrupt section: Timer event section 1128

Timer Event Sections. 1129

Defining the Scan Rate 1130

Defining the Phase 1131

Execution Order. 1134

Operating System 1135

Examples for Parameterization 1137

R.3 Interrupt section: I/O event section 1141

I/O Event Sections. 1142

Priority 1143

Runtime Error. 1144

R.4 Modules for interrupt sections 1146

EFBs for Interrupt Sections 1146

Appendix S Automatic Connection to the PLC 1147

Automatic Connection with Command Line Parameters (Modbus, Modbus +, TCP/IP) 1148

Automatic Connection with the CCLaunch Tool (Modbus Plus) 1151

Glossary 1157

Index 1187

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

⚡ The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

⚠ This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a potentially hazardous situation which, if not avoided, **can result in** death or serious injury.

⚠ **CAUTION**
CAUTION indicates a potentially hazardous situation which, if not avoided, **can result in** minor or moderate injury.

CAUTION
CAUTION, used without the safety alert symbol, indicates a potentially hazardous situation which, if not avoided, **can result in** equipment damage.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and the installation, and has received safety training to recognize and avoid the hazards involved.

About the Book

At a Glance

Document Scope

This user manual is intended to help you create a user program with Concept. It provides authoritative information on the individual program languages and on hardware configuration.

Validity Note

The documentation applies to Concept 2.6 for Microsoft Windows 98, Microsoft Windows 2000, Microsoft Windows XP and Microsoft Windows NT 4.x.

NOTE: Additional up-to-date tips can be found in the Concept README file.

Related Documents

Title of Documentation	Reference Number
Concept Installation Instructions	840 USE 502 00
Concept IEC Block Library	840 USE 504 00
Concept EFB User Manual	840 USE 505 00
Concept LL984 Block Library	840 USE 506 00

You can download these technical publications and other technical information from our website at www.schneider-electric.com.

User Comments

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

General description of Concept

1

Overview

This chapter contains a general description of Concept. It should provide an initial overview of Concept and its helper programs.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
1.1	General description of Concept	24
1.2	Programming	29

1.1 General description of Concept

Overview

This section describes the performance features of Concept and provides an overview of the hardware that may be programmed using Concept.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction	25
PLC hardware configuration	27
PLC Hardware Package Contents in Concept S, M and XL	28

Introduction

Operating System

Nowadays, a graphical user interface is a requirement for tasks of this kind. For this reason, Concept has been established as an MS Windows application. Concept can be operated in Windows 98, Windows 2000, Windows XP and Windows NT. These operating systems have the advantage that they are used all over the world. Therefore PC users have a basic knowledge of Windows technology and mouse operation. In addition to this all common monitors, graphic cards and printers can be used with MS Windows. As a user, you are not therefore tied to specific hardware configurations.

International Standard IEC 1131-3

For effective system configuration Concept offers a unified configuration environment in accordance with international standard regulations IEC 1131-3.

PLC Independence when Programming

The guiding principle behind the development of Concept was that all the system configuration procedures and all the editors should have the same look and feel. Most of the configuration steps, especially program creation, are designed independently of the PLC to be programmed.

Graphical Interface

The entire program is divided up into sections corresponding to the logic structure.

The Concept configuration tool enables objects (such as function blocks, steps, and transitions) to be selected, placed and moved easily in graphical form. Plausibility tests already take place in the SFC editor (Sequential Function Chart/ sequence language) during object placing, as most of the links between objects are generated automatically during placing. In the FBD editor (Function Block Diagram/Function Block language) and LD editor (Ladder Diagram), plausibility tests take place when blocks are linked. Unauthorized links, such as those between different data types have already been rejected during configuration. A plausibility test also takes place in the LL984 editor (Ladder Logic 984) during placing. In the IL editor (Instruction List) and ST editor (Structured Text) unauthorized instructions are identified via a colored outline. After the first successful program run, the program may be optimized in graphic terms by moving links, blocks or texts to improve the display.

Print

If desired the sections may be displayed with print preview information, in order to individually control pages of documentation. Signals receive an expansive designation with symbol names and comments. Unique notes on signal tracking are provided at the signal breaks. The individual block processing sequences from one section may be displayed and documented in the FBD editor.

Import/Export Functions

Sections from various projects can be combined as desired in another project using import/export functions.

It is also possible to convert the sections of one IEC programmer language into sections of another IEC programmer language.

Variables may be imported into and exported from the text using text delimited or FactoryLink format.

Runtime System

The runtime system on the PLC offers quick reactions to signal state process changes (short cycle time), Simulating signal transmitters *(see page 745)*, Online display *(see page 631)*, online parameter changes and online program changes.

Open Software Architecture

Concept possesses open software architecture to enable connection to external systems (e.g. for visualization) via standard interfaces.

Online Help

Special care was taken when developing the help function. The context sensitive Online help function *(see page 815)* provides support for every configuration situation just by clicking on the subject using the mouse or pressing the F1 key. Menu commands and dialogs are also context sensitive, as are, function blocks and hardware components of the individual PLC families.

PLC hardware configuration

Description

Concept is the unified projection tool for Quantum, Compact, Momentum and Atrium products.

Hardware components (for example CPU, program memory, input/output units etc.) can be specified before, during or after program creation.

This projection task can be performed both online (linked to the PLC) and locally (PC alone). Projection is supported by Concept, and only suggests valid combinations. Misprojection is therefore prevented. In online mode the projected hardware is tested for plausibility immediately and input errors are rejected.

After linking the programmer device (PC) to the PLC, a plausibility test is performed on the projected values (e.g. from the Variable Editor) using the actual hardware resources and if necessary an error message will appear.

PLC Hardware Package Contents in Concept S, M and XL

Description

PLC Hardware Package Contents in Concept S, M and XL:

Concept version	**contain Hardware**
Concept Vx.x **S**	Momentum
Concept Vx.x **M**	Compact, Momentum
Concept Vx.x **XL**	Atrium, Compact, Momentum, Quantum

1.2 Programming

Overview

This section provides an overview of the editors which are available in Concept.

What's in this Section?

This section contains the following topics:

Topic	Page
General information	30
Libraries	31
Editors	33
Online functions	37
Communication	38
Secure Application	39
Utility program	41

General information

At a Glance

As a solution for automatic control engineering tasks, Concept provides the following IEC 1131-3 compatible programming languages:

- Function Block language FBD (Function Block Diagram) *(see page 33)*,
- LD (Ladder Diagram) *(see page 34)*,
- Sequential language SFC (Sequential Function Chart) *(see page 34)*,
- Instruction List IL *(see page 34)* and
- Structured Text ST *(see page 35)*.

The Modsoft orientated language is also available

- Ladder Diagram LL984 (Ladder Logic) *(see page 35)*.

The IEC programming language (FBD, LD, SFC, ST and IL) basic elements are Functions and Function Blocks, which make up assembled logic units. Concept contains various Block libraries *(see page 31)* with predefined elementary functions/Function Blocks (EFBs). In order to locate the individual EFBs without difficulty, they are split into different groups according to their area of use.

For the Modsoft orientated programming language LL984, there is a Block library *(see page 31)* with Instructions available.

Sections

The control program is constructed from sections according to the logic structure. Only one programming language is used within a section.

Merging these sections makes up the entire control program and the automation device uses this to control the process. Any IEC sections (FBD, LD, SFC, IL, ST) may be mixed within the program. The LL984 sections are always edited as a block before the IEC sections.

Data types

A subset of Data types from the international standard IEC1131-3 is available.

In the Data type editor *(see page 36)* intrinsic data types can be derived from IEC data types.

Using variables

Variables for linking basic elements (objects) within a section are not usually necessary with the graphic programming languages FBD, LD, SFC and LL984, as these links are usually made graphically. (An additional link using variables is only necessary for incredibly complex sections.) Graphic links are managed by the system and therefore no projection requirement is created. The Variable Editor *(see page 36)* is used to project all other variables such as those for data transfer between various sections.

Libraries

At a Glance

For program creation Concept provides various block libraries with predefined Functions and Function Blocks.

There are 2 different types of block libraries:

- IEC library
  Block libraries for sections in the IEC programming languages (FBD, LD, SFC, IL and ST)
- LL984 Library
  Block library for sections in the Modsoft orientated programming language LL984

IEC library

The following IEC libraries are available for applications:

- **AKFEFB**
  This library contains the AKF/ALD EFBs, which are not covered by the IEC library.
- **ANA_IO**
  This library is for analog value processing.
- **COMM**
  This library is used for exchanging data between a PLC and another Modbus, Modbus Plus or Ethernet node.
- **CONT_CTL**
  This library is for projecting process-engineering servoloops. It contains controller, differential, integral, and polygon graph EFBs.
- **DIAGNOSTICS**
  This library is used to investigate the control program for misbehaviors. It contains action diagnostics, Reaction diagnostics, locking diagnostics, process prerequisite diagnostics, dynamic diagnostics and signal group monitoring EFBs.
- **EXPERTS**
  This library contains EFBs, which are necessary for using expert modules.
- **EXTENDED**
  This library contains useful supplements for different libraries. It has EFBs for creating average values, selecting maximum values, negating, triggering, converting, creating a polygon with 1st degree interpolation, edge recognizing, and for specifying an insensitive zone for control variables.
- **FUZZY**
  This library contains EFBs for fuzzy logic.
- **IEC**
  This library contains the EFBs defined in IEC 1131-3. It has for example EFBs for mathematical calculations, counters, timers etc.

- **LIB984**
  This library contains IEC 1131 compatible EFBs from the LL984 library, for example, EFBs for register transfer.
- **SYSTEM**
  This library contains EFBs for using system functions. It has EFBs for cycle time recognition, for various system cycle use, for SFC section control and for system status display.

LL984 Library

The LL984 library contains the LL984 editor instructions (blocks). It contains instructions for mathematical calculations, counters, timers, instructions for displaying system status, controller, differential and integral instructions and instructions for exchanging data between a PLC and another Modbus or Modbus Plus node.

Editors

At a Glance

When generating a section specify which programming language you are going to use.

The following editors are available for creating sections in the various programming languages:

- FBD editor (Function Block Language) *(see page 33)*
- LD editor (Ladder Diagram) *(see page 34)*
- SFC editor (Sequence language) *(see page 34)*
- IL editor (Instruction List) *(see page 34)*
- ST editor (Structured Text) *(see page 35)*
- LL984 editor (Modsoft orientated Ladder Logic) *(see page 35)*

The following editors are available for declaring variables, creating data types and displaying variables.

- the Variable Editor (for declaring variables), *(see page 36)*
- the reference data editor (for displaying and online changing of values) *(see page 36)* and
- the data type editor (for creating user specific data types) *(see page 36)*.

The following editors are available for creating user specific functions and Function Blocks:

- Concept DFB (for creating Derived Function Blocks and macros) *(see page 41)*
- Concept EFB (for creating user specific elementary functions and Function Blocks) *(see page 42)*

FBD editor

The FBD editor *(see page 213)* is used for graphic function plan programming according to IEC 1131-3.

Elementary functions, Elementary Function Blocks (EFBs) and Derived Function Blocks (DFBs) are connected with signals (variables) onto FBD sections for the function plan. The size of a FBD section is 23 lines and 30 columns.

EFBs are equipped with a fixed or variable number of input variables and may be placed anywhere on the section. Variables and EFBs may have comments separately added to them, column layouts on a section may be commented on anywhere using text boxes. All EFBs may be performed conditionally or unconditionally.

All the EFBs are divided into function- and use-orientated libraries in various groups, to make them easier to locate.

LD editor

The LD editor *(see page 239)* is used for graphic ladder programming according to IEC 1131-3.

Contacts and coils are connected to the Ladder Diagram in LD sections using signals (variables).

The size of a FBD section is 23 lines and 30 columns.

Furthermore, the elementary functions and Function Blocks (EFBs), which are named in the FBD editor, the Derived Function Blocks (DFBs) and User Defined Function Blocks (UDFBs) may also be bound in the ladder diagram (see *FBD editor, page 33*).

The structure of a LD section corresponds to a rung for relay switching. The left power rail is located on its left-hand side. This left power rail corresponds to the phase (L ladder) of a rung. With LD programming, in the same way as in a rung, only the LD objects (contacts, coils) which are linked to a power supply, that is to say connected with the left power rail, are "processed". The right power rail, which corresponds to the neutral ladder, is not shown optically. However, all coils and EFB outputs are linked with it internally and this creates a power flow.

SFC editor

The SFC editor *(see page 271)* is used to graphically program an IEC 1131-3 compatible sequential control.

The SFC elements are connected in a SFC section to one of the sequential controls corresponding to the task setting. The size of a SFC section is 32 lines and 200 lines.

The following sequential control programming objects are available in Concept.

- Step (including actions and action sections)
- Transition (including transition section)
- Alternative branch and merge
- Parallel branch and merge
- Jump
- Connection

Simple diagnostics monitoring functions are already integrated in the steps.

IL editor

The IL editor *(see page 321)* is used for programming IEC 1131-3 compatible instruction lists.

Existing IL instructions, elementary functions and Elementary Function Blocks (EFBs), and Derived Function Blocks (DFBs) are written in series in text form in IL sections from operators (commands) and operands (signals, variables).

When the program is entered, all the standard Windows services and some additional commands for text-processing are available. The size of an IL section is 64 Kbyte maximum.

The following instruction list programming operators are available in Concept:
- Logic (AND, OR etc.)
- Arithmetic (ADD, SUB, MUL, DIV, …)
- Comparative (EQ, GT, LT, …)
- Jumps (JMP, … conditional/unconditional)
- EFB call (CAL , … conditional/unconditional)

IL programming is done in text form. When text is entered, all the standard Windows services for text-processing are available. The IL editor also contains some further commands for text-processing.

A spell check is performed immediately after text has been entered (instructions, key words, separators), highlighting errors with a colored outline.

ST editor

The ST editor *(see page 393)* is used for programming IEC 1131-3 structured text.

Existing ST statements, elementary functions and Elementary Function Blocks (EFBs), and Derived Function Blocks (DFBs) are written in text form in IL sections by printing (operator lists) and operands (signals, variables).

When the program is entered, all the standard Windows services and some additional commands for text-processing are available. The size of a ST section is 64 Kbyte maximum.

The following structured text programming statements and operators are available in Concept:
- conditional/unconditional statement execution (IF, ELSIF, ELSE, …)
- conditional/unconditional loop execution (WHILE, REPEAT)
- Mathematical, comparative, and logic operators
- conditional/unconditional EFB call

ST programming is done in text form. When text is entered, all the standard Windows services for text-processing are available. The ST editor also contains some further commands for text-processing.

A spell check is performed immediately after text has been entered (instructions, key words, separators), highlighting errors with a colored outline.

LL984 editor

Using the Modsoft orientated LL984-Editor *(see page 455)* (Ladder Diagram 984), instructions, contacts, coils and signals (variables) are connected to a ladder diagram. Instructions, contacts, coils and variables may be commented on.

The structure of a LL984 section corresponds to a rung for relay switching. The left power rail is located on its left-hand side, but it is not visually displayed. This left power rail corresponds to the phase (L ladder) of a rung. With LL984 programming, in the same way as in a rung, only the LL984 objects (instructions, contacts, coils) connected to a power supply, i.e. connected to the left power rail, are "processed". The right power rail, which corresponds to the neutral ladder is not visually displayed either. However, all coils and instruction outputs are linked with it internally and this creates a power flow.

Concept has various predefined instructions for ladder programming using LL984. These may be found in the block library LL984. Additional instructions for special applications are available as loadables and may be loaded at a later time.

Variable Editor

The Variable Editor *(see page 543)* is used to declare and comment on all necessary symbolic signal names (variables). Only declared variables may be used in Concept programs.

A data type must be assigned to each symbolic signal name! If this variable is assigned a reference address, a Located variable (without reference address = Unlocated variable) is received. An initial value may also be provided for each variable, which will be transferred into the PLC during the first load.

Data type editor (DDT editor)

The Data type editor *(see page 565)* may be used to define specific Derived Data Types (Derived Data Type = DDT).

Derived Data Types combine several Elementary data types (BOOL, WORD, …) in one data record. It is not only the same data types which may be combined as ARRAY, but also various data types may be combined as STRUCT. In Concept, a number of Derived Data Types are already available, which for instance may be used for DFBs.

DDTs appear in DFBs or EFBs only as a connection, i.e. for instance in FBD a variable input is only necessary in the block. It is thus recommended that frequently recurring groups of elementary data types (and also DDTs) be defined as DDTs, in order to improve accessibility of an application.

The definition appears in text form, and all the standard Windows services and some additional commands for text-processing are available. The size of a data type file is 64 Kbyte maximum.

Reference data editor

The Reference data editor *(see page 595)* may be used in online mode to display the variable value, to force variables and to set variables. There is also the possibility of separating variables from the process. Inputs may be saved in a data file and be reused.

Online functions

Available online functions

After the programming device has been linked to the PLC, a range of online Startup and maintenance functions become available.

- the program on the programming device is compared with the program on the PLC
- the PLC can be started and stopped
- Object information is displayed
- Programs can be loaded, sections can be changed online and loaded
- Variable values can be entered online
- Animation mode shows the program with its current signal states

Operating and monitoring

Declaration of special operating and monitoring variables is not necessary in Concept. The variables to be visualized can be identified as such in the Variable Editor and then be exported into a ModLink or FactoryLink configuration data file. This data file can be used for visualizing.

Communication

Description

Communication between the PLC and another Modbus-, Modbus Plus-, SY/MAX-Ethernet or TCIP/IP Ethernet node is projected using IEC languages (FBD, LD, SFC, ST, IL) with the EFBs from the block library COMM. The instruction MSTR may be used with the programming language LL984 to construct these communications.

A peer to peer transfer of register contents is possible using the peer cop, independent of these blocks/instructions.

Communication is projected between the PLC and the decentralized I/O via the INTERBUS by simply entering the NOA module in the component list and loading a loadable (ULEX).

Communication is projected between the programming device and a PLC via Ethernet by simply entering and parametering the appropriate couple module in the component list.

Secure Application

At a Glance

In several areas of industry, the need for security demands regulated access to PLCs, recording program changes and archiving those recordings. Following a standardized procedure ensure that records may not be falsified. To enable these requirements, new features have been implemented in Concept that ensure secure application. To guarantee that all of these parameters are defined, the user can activate the **Secure Application** check box in the **Project** →**Project Properties** dialog. Concept will then ensure that all of these parameters are set and that their contents remain valid. The project is then indicated as being a secure application, and this information is included in the information that is downloaded to the PLC.

Secure Application

The secure application is defined in the **Project** →**Project Properties** dialog by activating the **Secure Application** check box. These settings are then exported, imported, read and loaded to the PLC.

NOTE: When the secure application is activated, a NOT EQUAL status is generated and required reloading to the PLC. Unchecking the check box also creates a NOT EQUAL status so that loading is again required as well. If Concept is connected to a PLC that is already defined with the "Secure Application" setting, the setting is automatically accepted in Concept in case of upload the controller.

The log file is stored in the Concept directory and has the name of the current date (YEARMONTHDAY.ENC, e.g. 20020723.ENC). The path of the log file can be defined in dialog **Common Preferences**. If no path is defined then Concept uses the default log path (Concept directory, e.g. C:\CONCEPT).

Among other things, logging write-access to the PLC can record the following data:
- Section name
- EFB/DFB Instance name, FB Type name
- Pin Name
- [Variable name] [Literal] [Address]
- Old value
- New value
- User name (if the Concept (Login) password is activated in Concept Security)
- Data and Time (see also*Address format in LOG file [Logging], page 1116*)

Requirements

The secure application can only be activated if the following prerequisites are met:
- can only be used with 140 CPU 434 12A or 140 CPU 534 14A/B
- at least one IEC section (if no IEC section exists then the download is aborted.)
- Offline mode (**Online** →**Disconnect...**)
- Supervisor Rights (see Concept under **Help** →**About...** →**Current User:**)

Activation Combination for Secure Application

Various Activation Combinations for Secure Application:

"Secure Application" activated in Concept	"Secure Application" loaded to PLC	Reaction to connection with the PLC
Not activated	Not activated	Normal operation without secure application
Not activated	Activated	When uploading, the **Secure Application** check box is activated in Concept and encrypted logging is activated.
Activated	Not activated	Download required because the status is NOT EQUAL.
Activated	Activated	Normal operation with secure application (e.g. encrypted logging).

Reading the Encrypted Log File

To read the encrypted log file, the View tool is opened automatically in the **View Logfile** dialog.

NOTE: If an encrypted log file has been improperly modified in any way, the log is decoded as much as is possible, and the lines that have been modified will remain unreadable. The first line will contain the message: "This log file has been modified".

Utility program

At a Glance

In addition to Concept the following range of utility programs are available:

- Concept DFB
- Concept EFB
- Concept SIM (16 bit)
- Concept PLCSIM32 (32 bit)
- Concept Security
- Concept WinLoader
- Concept Converter
- Concept ModConnect

Concept DFB

Concept DFB is used to create DFBs (Derived Function Blocks) *(see page 481)* and Macros *(see page 521)*.

DFBs (Derived Function Blocks)

DFBs can be used for setting both the structure and the hierarchy of a program. In programming terms, a DFB represents a subroutine.

DFBs can be created in the programming languages FBD, LD, IL, and ST. In Concept, DFBs can be called up in any programming language, regardless of the programming language they were created in. One or several existing DFBs can be called up within one DFB, with the called-up DFBs themselves able to call up one or several DFBs.

Macros

Macros are used to duplicate frequently used sections and networks (including their logic, variables and variable declaration).

Macros have the following properties:

- Macros can only be created in the programming language FBD.
- Macros only contain one section.
- Macros can contain a section of any complexity.
- In programming terms, there is no difference between an instanced macro, i.e. a macro inserted into a section and a conventionally created section.
- It is possible to call up DFBs in a macro.
- It is possible to declare macro-specific variables for the macro.
- It is possible to use data structures specific to the macro
- Automatic transfer of the variables declared in the macro.
- Initial values are possible for the macro variables.
- It is possible to instance a macro many times in the entire program with different variables.
- Section names, variable names and data structure names can contain the character ~ as an exchange marking.

Concept EFB

The optional tool Concept EFB can be used to generate, in C++ programming language, your own application specific Functions and Function Blocks (EFBs) and to integrate them in the form of libraries with groups in your version of Concept.

The operating rules for these user-defined blocks (UDFBs) are identical to those for standard EFBs.

It is, for instance, recommended that complex program parts with a high number of calls and program parts, whose solution is to remain hidden from the user, e.g. special technology objects etc. be generated using Concept EFB.

NOTE: Concept EFB is not included as part of the Concept package and may be ordered in addition.

Concept SIM (16 bit)

The 16 bit simulator Concept SIM *(see page 746)* is available for simulating a PLC, i.e. to test your user program online without hardware. Concept SIM simulates a coupled PLC via Modbus Plus.

NOTE: The simulator is only available for the IEC languages (FBD, SFC, LD, IL and ST).

Concept PLCSIM (32 bit)

The 32 bit simulator Concept PLCSIM32 *(see page 748)* is available for simulating a PLC, i.e. to test your user program online without hardware. Concept PLCSIM32 simulates a PLC coupled via TCP/IP, where the signal states of the I/O modules can also be simulated. Up to 5 programming devices can be coupled to the simulated PLC at the same time.

NOTE: The simulator is only available for the IEC languages (FBD, SFC, LD, IL and ST).

Concept Security

Concept Security *(see page 757)* can be used to assign access. Access signifies that the function of Concept and its utility programs is limited depending on the user.

The access defined for one user is applicable to all Concept installation projects. A maximum of 128 users may be defined.

Concept Converter

Projects, DFBs, macros, and data structures (Derived Data Types), created for an earlier version of Concept, can be converted without hassle to work in the current version of concept in the Concept Converter *(see page 991)*.

Concept EXECLoader

The Concept EXECLoader can be used to load Exec data files onto the PLC.

Concept ModConnect

Concept-ModConnect *(see page 995)* can be used to extend the configurator for new (specific) I/O modules.

New Performance Attributes of Concept 2.6 in Comparison with Concept 2.5

2

Overview

This Chapter describes the new performance attributes of Concept 2.6 in comparison with Concept 2.5.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
New Performance Attributes of Concept 2.6 Compared with Concept 2.5	46
New performance attributes of Concept 2.6 SR2 in comparison with Concept 2.6 SR1	52
New performance attributes of Concept 2.6 SR3 in comparison with Concept 2.6 SR2	55

New Performance Attributes of Concept 2.6 Compared with Concept 2.5

Highlights

New general performance attributes:
- **Interrupt sections**
- **Global variables**
- **Security features**

New EFBs

New EFBs in the SYSTEM library:

New EFBs	Description
I_LOCK	Disable all interrupt sections
I_UNLOCK	Enable all interrupt sections
I_MOVE	Interrupt protected assignment
ISECT_OFF	Disable specific interrupt sections
ISECT_ON	Unlock a specific interrupt section
ISECT_STAT	Interrupt section status
PRJ_VERS	States project name and version
GET_IEC_INF	Read IEC status flags
RES_IEC_INF	Reset IEC status flags

New EFBs in the COMM library:

New EFBs	Description
PORTSTAT	States Modbus Port status

Start Concept

New features when starting Concept:

New performance attributes	Description
Automatic connection to every **desired** PLC	Startup using the Concept Project Symbol creates automatic connection to any desired PLC. This connection is defined by the Command line parameter *(see page 1148)*.
When starting Concept using the CCLaunch tool, a connection is made to every **desired** PLC	In large networks, a topology file is created and is then used in the CCLaunch tool. You can use this to create a complete MB+ Routing path *(see page 1151)*, which then creates a connection to the PLC automatically.

New performance attributes	Description
Displays list of previously opened Projects/DFBs	When starting Concept a list of previously opened Projects/DFBs (max. 4) is displayed in the **File** main menu.
Archive content display	When unpacking an archived project, all archived files are shown first.

Animation

12 different color schemes for animation in the FBD, IL, ST, SFC and LD editors:

New performance attributes	Description

CONCEPT.INI:
`[Colors]`
`AnimationColors= (0-12)`
	Defines the color scheme for online animation in all editors.

Reference data editor

New feature in the reference data editor:

New performance attributes	Description
Address format IEC (QW0000X)	The IEC (QW0000X) address format can be displayed.

Online functions

New online features:

New performance attributes	Description
Quantum password protection	Quantum PLC is write protected by entering a password.
Event sections	Online diagnostics are displayed for Interrupt sections.
Event viewer	Error descriptions can be defined in a project specific INI file *(see page 1122)* that should appear in the event viewer (**Online** →**Online events...**).

Message window

New performance attributes in the Windows menu:

New performance attributes	Description
Save messages	After messages are displayed they can be saved to file using the **Save Messages...** (main menu **Window**) menu command.

New CPU

New CPU:

PLC family	Description
Atrium	CPU 180-CCO-241-11

New Module

New Quantum module:

Module	Description
140-NOE-771-01	Ethernet module without Hot Standby features.
140-NOE-771-11	Ethernet module (Factory Cast) without Hot Standby features.
140-CPS-114-20	Power supply module
140-CPS-124-20	Power supply module
140-NOG-111-00	1/SFB Master module
140-NWM-100 00	Ethernet module (Factory Cast HMI)

New Momentum module:

Module	Description
170-ANR-120-91	Analog/Digital Input/Output module

Project Browser

New features in the Project browser:

New performance attributes	Description
Display interrupt sections	When I/O event sections and Timer event sections are used, they are displayed in the Project browser structure.
Show detailed view	The Project browser window is split vertically, and a second window displays the substructure (e.g. DFBs, Transitions sections, etc.) of the selected elements in a structure tree.

Analyze section

New features when analyzing sections:

New performance attributes	Description
Analyze interrupt sections	There is now an additional analysis for Interrupt sections.
Analyzing global variables in DFBs	There is an analysis for global variables in DFBs.

DFB

New features for DFB programming:

New performance attributes	Description
Located variables	Located variables are permitted in DFBs when the option in the **IEC Extensions** dialog box is enabled. Global variables can be created throughout the program with located variables in DFBs.

Data types

New features for DFB programming:

New performance attributes	Description
View comments for data structure elements	Comments for data type components defined in data type files (*.ddt, *.dty) are displayed in: ● Editors status line ● Variables editor for the definition of initial values ● Inspect Animation field
Extended Data Type Definition (larger than 64 Kbytes), page 572	The 64 kb restriction is not imposed for local data type definition with the introduction of unlocated Include files.

Configuration

New features in the Configurator:

New performance attributes	Description
1/SFB Coupler configuration	Required to provide support for the A500/A350 I/O module. Extended I/O range up to 160 input/output words.
Quantum security parameter	The following parameters can be defined in the new dialog box (submenu of the **Config. Extensions**): ● Secure data area ● Network write restrictions ● Enable the Auto-Logout option
Interbus configuration with Atrium	The Interbus configuration is done with Atrium CPUs 180 CCO 241 01 (= 1 INTERBUS) and 180 CCO 241 11 (= 2 INTERBUS).

Logging (*.LOG, *.ENC)

New features for DFB logging:

New performance attributes	Description
Additional contents	When logging PLC write access, modifications made to variable and literal values are displayed in addition.
New Date/Time format	By activating the check box **Universal Date Format** in dialog **Common Preferences** (setting also affects the CONCEPT.INI file) the format can be changed. The month is then stated within Concept with 3 characters and in English. **Example:** 24-Dec-2002 14:46:24
Encrypting the log	By activating the check box **Encrypt Logfile** in dialog **Common Preferences** (or indirectly using the check box **Secure Application** in dialog **Project Properties**) login the write access to the PLC will be encrypted. The encrypted file contains the file extension *.ENC.

Secure Application

New features for a secured application:

New performance attributes	Description
Application backup	If you activate the check box in the **Project →Project Properties** dialog box, program modifications are automatically logged and encrypted in a *.ENC file. These settings can be loaded using Export/Import and transferred to the PLC.

New Tools

New Tools for Concept:

New Tool	**Description**
CCLaunch	This tool is used for making an automatic connection *(see page 1151)* with a PLC in a large network.
View Tool	This tool allows you to view encoded LOG files (*.ENC). It is started automatically with menu instruction **View Logfile** if log encrypting has been activated.

New performance attributes of Concept 2.6 SR2 in comparison with Concept 2.6 SR1

New EFBs

New EFBs in the IEC library:

New EFBs	Description
CMPR	Compares the Bit pattern of Matrix A to that of Matrix B.
MBIT with pointer	Changes the bit position in a data matrix.
SEARCH	Searches the register in a source table for a specific bit pattern.
SENS with pointer	Checks the query value of a specific bit position in a data matrix.
XXOR	Performs a Boolean Exclusive-OR-Operation with the bit patterns of the source and target matrix.

Search/Replacement of FFBs

New features when searching for/replacing FFBs:

New feature	Description
FFB type is replaced in all sections (only for DFBs)	In the dialog box **Replace FFB Type** by activating the new check box **Replace in all sections** the selected FFB type can be replaced in all sections (only for DFBs).

Create a new project

New features when generating a new project:

New feature	Description
Specify project path when generating a new project	When generating a new project (**File →New Project**) you can define a new path or accept the standard path again.

New options in the upload and loading dialog box

New options in the upload and loading dialog box:

New features	Description
New check boxes in the dialog box **Load into the PLC**: ● State RAM + Initial Values ● Only state RAM	By activating the check box **State RAM + Initial Values** at first all initial values of the Located 4x-Variables are copied from the Variable Editor into the state RAM mirror. Then, the initial values and all blocked 0x and 1x-I/O-bits are loaded from the state RAM mirror into the PLC. By activating the check box **State RAM Only** the initial values of the Located 4x-Variables and all blocked 0x and 1x I/O bits are loaded from the state RAM mirror into the PLC.
New check boxes in the dialog box **PLC Upload**: ● Upload State RAM + Initial Values ● Only upload State RAM	By activating the check box **Upload State RAM + Initial Values** at first all Located 0x-, 1x, and 4x-values are read from the PLC and saved in the state RAM mirror. Then, the initial values of the 4x-variables are overwritten with the value from the state RAM mirror. With the activation of the check box **Only read state RAM** all Located 0x-, 1x- and 4x-values are read from the SPS, and saved in the state RAM mirror.

INI files

New settings in the CONCEPT.INI:

New Settings	Description
Define overwriting of the uploaded state RAM values	In the line [RDE] of the CONCEPT.INI you can define that uploaded state RAM values are not overwritten by online operations in the RDE.
Define start of the RDE-Animation	In the line [RDE] of the CONCEPT.INI you can define that the RDE animation is automatically started when opening a table.
Exclusion of all or global DFBs from Online-Backup	In the line [Backup] of the CONCEPT.INI you can define that after the Online-Backup the directories "DFB" and/or "DFB.GLB" are not present in the backup directory.

New settings in the Projectname.INI:

New Setting	Description
Define path and backup files	In the line [Backup] of Projectname.INI, you can output a Batch-file (EXE-file) for the Online-Backup-Operation, by which you perform additional backups e.g. for another PC.

Multiple Address Assignment

New feature for multiple address assignment:

New feature	Description
Cleaning up multiple assignment of a single address by different variable names	In the dialog box **Multiple Address Assignments** variable names that are all assigned to the same address are replaced or renamed. In the end, only one variable name is assigned to this address.

New performance attributes of Concept 2.6 SR3 in comparison with Concept 2.6 SR2

New menu command

New menu command:

New menu command	Description
Options →Tools	Use this menu command to open a menu to execute additional applications or help programs.

Project structure

3

Overview

This chapter describes the structure of projects in Concept.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Project Structure and Processing	58
Programs	64
Sections	68
Configuration data	73

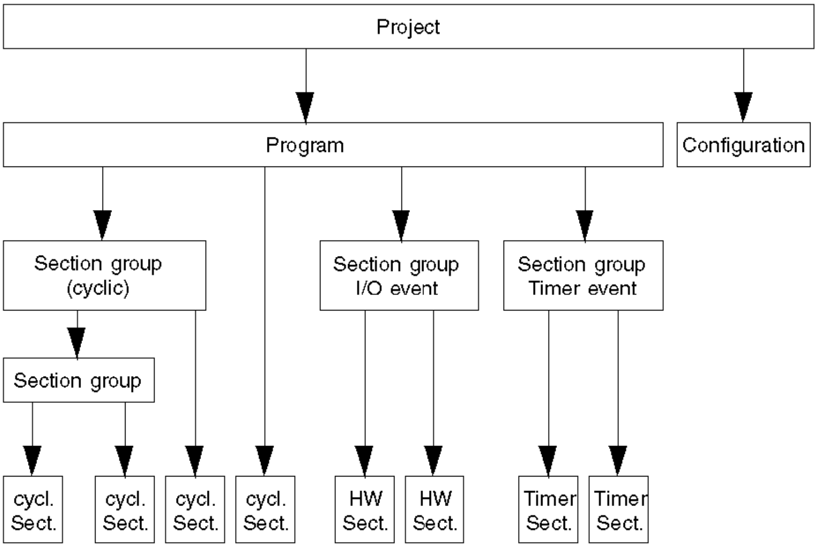
Project Structure and Processing

Structure of a project

The creation of a PLC program with Concept is carried out hierarchically in a project using PLC configuration *(see page 73)* and Program *(see page 64)*. The program is divided into section groups and Sections *(see page 68)*.

The PLC configuration and required program parts can be created in any order within a project (top down or bottom up).

Structure of a project:



Processing an IEC/LL984 project

This table describes the processing of a LL984/IEC project (Quantum):

Step	Logic processor	I/O processor
1	Overhead, e.g. communication with NOM, NOE etc.	-
2	Executing LL984 segment 1	Writing outputs calculated in segment n
		Reading inputs required in segment 2
3	Executing LL984 segment 2	Writing outputs calculated in segment 1
		Reading inputs required in segment 3

Step	Logic processor	I/O processor
4	Executing LL984 segment 3	Writing outputs calculated in segment 2
		Reading inputs required in segment 4
...	...	...
n	Executing LL984 segment n (n =< 32)	Writing outputs calculated in segment n-1
		Reading inputs required in segment 1
n+1	Executing IEC section 1	-
n+2	Executing IEC section 2	-
n+3	Executing IEC section 3	-
	..	-
m	Executing IEC section n (n =< 1600) and back to stage 1	-

1 The overhead is executed in this stage (e.g. communication with the coupling modules NOM, NOE).

2 - 4 In these stages, the logic for the LL984 sections is executed by the logic processor in segments 1-3 (corresponding to the settings in the Segment scheduler *(see page 117)*).
At the same time the I/O processor transfers the output values calculated in the respective previous segment to the hardware and the hardware reads the input values required for the next respective segment.

n In this step, the logic processor in segment n runs the LL984 sections logic.
At the same time the I/O processor transfers the output values calculated in the previous segment to the hardware and the hardware reads the input values required for segment 1.
Note: The output values calculated in this segment are only executed on next execution of stage 2, i.e. after the IEC logic and the overhead have been processed. Therefore no time critical logic should be executed in this segment.

n+1 - m The logic processor runs the IEC sections logic in these steps.
It then "jumps back" to stage 1.
Note: No hardware signals are read or written. The values calculated/read in stages 2 to n are used exclusively. The outputs calculated in these stages are transferred in stages 2 to n (corresponding to the settings in the segment scheduler).

Processing a LL984 project

This table describes the processing of a LL984 project (Quantum):

Step	Logic processor	I/O processor
1	Overhead, e.g. communication with NOM, NOE etc.	-
2	Executing LL984 segment 1	Writing outputs calculated in segment n
		Reading inputs required in segment 2
3	Executing LL984 segment 2	Writing outputs calculated in segment 1
		Reading inputs required in segment 3
4	Executing LL984 segment 3	Writing outputs calculated in segment 2
		Reading inputs required in segment 4
...	...	...
n	Executing LL984 segment n (n =< 32) and back to stage 1	Writing outputs calculated in segment n-1
		Reading inputs required in segment 1

1 The overhead is executed in this stage (e.g. communication with the coupling modules NOM, NOE).

2 - 4 In these stages, the logic for the LL984 sections is executed by the logic processor in segments 1-3 (corresponding to the settings in the Segment scheduler *(see page 117)*).
At the same time the I/O processor transfers the output values calculated in the respective previous segment to the hardware and the hardware reads the input values required for the next respective segment.

n In this step, the logic processor in segment n runs the LL984 sections logic.
At the same time the I/O processor transfers the output values calculated in the previous segment to the hardware and the hardware reads the input values required for segment 1.
It then "jumps back" to stage 1.
Note: The output values calculated in this segment are only processed the next time stage 2 is completed, i.e. after the overhead has been processed. Therefore no time critical logic should be executed in this segment.

Processing an IEC project

This table describes the processing of an IEC project (Quantum):

Step	Logic processor	I/O processor
1	Overhead, e.g. communication with NOM, NOE etc.	-

Step	Logic processor	I/O processor
2	-	Writing outputs allocated to segment 1
		Reading inputs allocated to segment 1
3	-	Writing outputs allocated to segment 2
		Reading inputs allocated to segment 2
4	-	Writing outputs allocated to segment 3
		Reading inputs allocated to segment 3
...	...	...
n	-	Writing outputs allocated to segment n (n =< 32)
		Reading inputs allocated to segment n (n =< 32)
n+1	Executing IEC section 1	-
n+2	Executing IEC section 2	-
n+3	Executing IEC section 3	-
	..	-
m	Executing IEC section n (n =< 1600) and back to stage 1	-

1 The overhead is executed in this stage (e.g. communication with the coupling modules NOM, NOE).

2 - n The hardware signals from the allocated modules respective segments are written and read by the I/O processor in these stages (corresponding to the settings in the Segment scheduler *(see page 117)*).

n+1 - m The logic processor runs the IEC sections logic in these steps.
It then "Returns" to stage 1.
Note: No hardware signals are read or written. The values read in stage 2 to n are used exclusively. The outputs calculated in these stages are transferred in stages 2 to n (corresponding to the settings in the Segment manager).

Processing an IEC project

This table describes the processing of an IEC project (Quantum):

Step	Logic processor	I/O processor
1	Overhead, e.g. communication with NOM, NOE etc.	-
2	-	Writing outputs allocated to segment 1
		Reading inputs allocated to segment 1
3	-	Writing outputs allocated to segment 2
		Reading inputs allocated to segment 2

Step	Logic processor	I/O processor
4	-	Writing outputs allocated to segment 3
		Reading inputs allocated to segment 3
HE1	1. I/O event section, spontaneous execution, when Hardware Interrupt occurs	-
HE2	2. I/O event section, spontaneous execution, when Hardware Interrupt occurs	-
...	...	...
HE64	64. (last) I/O event section, spontaneous execution, when Hardware Interrupt occurs	-
TE1	1. Timer event section, only executed when time interrupt occurs	-
TE2	2. Timer event section, only executed when time interrupt occurs	-
...	...	...
TE16	16. Timer event section, only executed when time interrupt occurs	-
...	...	...
n	-	Writing outputs allocated to segment n (n =< 32)
		Reading inputs allocated to segment n (n =< 32)
n+1	Executing IEC section 1 (cyclically)	-
n+2	Executing IEC section 2 (cyclically)	-
n+3	Executing IEC section 3 (cyclically)	-
	..	-
m	Executing IEC section n (n =< 1600) and return to stage 1	-

1 The overhead is executed in this stage (e.g. communication with the coupling modules NOM, NOE).

2 - n The hardware signals from the allocated modules respective segments are written and read by the I/O processor in these stages (corresponding to the settings in the Segment scheduler *(see page 117)*).

n+1 - m The logic processor processes the IEC sections logic in these steps. It then "Returns" to stage 1.

Note: No hardware signals are read or written. The values read in stage 2 to n are used exclusively. The outputs calculated in these stages are transferred in stages 2 to n (corresponding to the settings in the Segment scheduler).

HE1 - HE64 If a hardware interrupt signal specially assigned to a section changes its value according to its parameter configuration, the cyclical processing and if necessary the processing of a Timer event section is immediately stopped and returned to the I/O event section. Once all event sections (and Timer event sections) are processed, the cyclical processing is continued at the point where the interrupt occurred. (See also chapter "*I/O Event Sections, page 1142*")

TE1 - TE16 When a specially configured Timer interrupt signal for a section occurs, cyclical processing is immediately stopped and jumps to the Timer event section. Once Timer event sections are processed, the cyclical processing is continued at the point where the interrupt occurred as long as there are no further instructions for Timer event sections. (See also chapter "*Timer Event Sections, page 1129*")

Programs

Structure of a program

A program consists of one or more Sections *(see page 68)* or section groups. Section groups can contain sections and other section groups. Section groups can be created exclusively and filled using **Project →Project browser**. Sections describe the entire systems mode of operating.

Moreover the variables, constants, literals and direct addresses are managed within the program.

Variables

Variables are used to exchange data within a section, between several sections and between the program and the PLC.

Variables are declared using the menu command **Project →Variable declaration**. If the variable with this function is assigned an address, it is called a Located variable. If the variable has no address assigned to it, it is called an Unlocated variable. If the variable is assigned with a derived data type, it is called a Multi-element variable.

There are also constants and literals.

The following table provides an overview of the various types of variables:

Variable type	Description
Located variables	Located variables are allocated a State RAM address (reference address 0x, 1x, 3x,4x). The value of this variable is saved in the State RAM and can be changed online using the Reference data editor. These variables can be addressed using their symbolic names or using their reference address. All PLC inputs and outputs are connected to the State RAM. The program can only access peripheral signals attached to the PLC via located variables. Access from external pages via Modbus or Modbus Plus interfaces of the PLC, e.g. from visualization systems can be made using located variables.
Unlocated variables	Unlocated variables are not assigned a State RAM addresses. They therefore do not occupy any State RAM addresses. The value of this variable is saved internally in the system and can be changed using the Reference data editor. These variables are only addressed using their symbolic names. Signals requiring no peripheral access, e.g. intermediate results, system tags etc, should primarily be declared as unlocated variables.

Variable type	Description
Multi element variables	A variable which is assigned a Derived data type. A distinction is made here between Structured variables and Array variables.
Structured variables	Variables to which a Derived data type defined using a STRUCT (structure) is assigned. A structure is a collection of data elements with generally different data types (Elementary data types and/or Derived data types).
Array variables	A variable which is assigned a defined data type with the key word ARRAY. An array is a collection of data elements with the same data type.

Variable start behavior

In start behavior of PLCs there is a distinction between cold restarts and warm restarts:

- **Cold restart**
  Following a cold restart (loading the program with **Online** →**Download**) all variables (irrespective of type) are set to "0" or their initial value if available.
- **Warm restart**
  In a warm restart (stopping and starting the program or **Online** →**Download changes**) different start behaviors are valid for located variables/direct addresses and unlocated variables:
  - **Located variables/direct addresses**
    In a warm restart all 0x, 1x and 3x registers are set to "0" or their initial value if available.
    The buffered coils are an exception to this. Buffered coils retain their current value (storage behavior).
    4x registers retain their current value (storage behavior).
  - **Unlocated variables**
    In a warm restart all unlocated variables retain their current value (storing behavior).

This varying behavior in a warm restart leads to peculiarities in the warm restart behavior of set and reset functions.

- **Set and Reset in LD and IL**
  Warm restart behavior is dependent on the variable type used (storage behavior in use of unlocated variables; non storage behavior in use of located variables/direct addresses)
- **SR and RS Function Blocks in FBD, LD, IL and ST**
  These function blocks work with internal unlocated variables and therefore always have a storage behavior.

Constant variables

Constants are unlocated variables assigned a value, which cannot be modified by the logic program (read only).

Literals (values)

Literals are used to describe FFB inputs, and transition conditions etc using direct values. These values cannot be overwritten by the program logic (read only).

The values of literals can be changed online.

There are two different types of literal; generic and standardized.

The following table provides an overview of the various types of literals:

Literal	Description
Generic literals	If the literal's data type is not relevant, simply specify the value for the literal. In this case, Concept automatically assigns a suitable data type to the literal.
Standardized literals	If you would like to manually determine a literal's data type, this may be done using the following construction: "Data type name"#"Literal value" For example INT#15 (Data type: Integer, value: 15), BYTE#00001111 (Data type: Byte, value: 00001111) REAL#23.0 (Data type: Real, value: 23.0) To assign the data type **REAL** the value may also be specified in the following manner: 23.0. Entering a comma will automatically assign the data type REAL.

Direct addresses

Direct addresses are memory ranges in the PLC. They are located in the State RAM and can be assigned Input/Output modules.

Direct addresses can be entered or displayed in various formats. The display format is specified in the dialog box **Options** →**Preferences** →**Common**. Setting the display format has no impact on the entry format, i.e. direct addresses can be entered in any format.

The following address formats are possible:

- **Standard format (400001)**
  The five character address comes directly after the first digit (the Reference).
- **Separator format (4:00001)**
  The first digit (the Reference) is separated from the following five-character address by a colon (:).
- **Compact format (4:1)**
  The first digit (the Reference) is separated from the following address by a colon (:), and the leading zeros of the address are not given.
- **IEC format (QW1)**
  In first place, there is an IEC identifier, followed by the five-character address.
  - %0x12345 = %Q12345
  - %1x12345 = %I12345
  - %3x12345 = %IW12345
  - %4x12345 = %QW12345

The values of direct address can be modified online using the Reference data editor *(see page 595)*.

Start behavior of digital outputs

Outputs that are assigned 0x registers are deleted during PLC startup. Digital outputs that assigned 4x registers keep their current value when the PLC is stopped or started.

Sections

Introduction

A program consists of one or more sections. A section describes the mode of functioning of a systems technological unit (for example a motor).

Each section has its own document window in Concept. For overview purposes it is useful to divide a very large section into several small ones. The scroll bar is used to move within a section.

The page break can be made visible for each section, so that the page format can be monitored when programming. In this way, a readable printout of the section is assured.

Section types

There are three different types of sections in Concept provided for Quantum processing.

- **Cyclical section** are executed in every program cycle. The reaction time depends on the cycle time and is a minimum of one cycle and maximum of two cycles.
- **I/O event sections** are not executed cyclically, but are started and processed spontaneously when a specially assigned Interrupt signal value changes state (corresponding to the setting in the Configurator and Section properties).
  The 140-HLI-340-00 module provides 16 Interrupt inputs. The local backplane has space for a maximum of 4 HLI modules.
  The reaction time to an I/O event generally depends on the process duration of the EFBs to be processed in the section as well as the transition times.
- **Timer event sections** are started and processed in precise user defined intervals.
  The time intervals are defined in multiples of 1ms and a Phase in the Section properties for Timer Event Sections dialog box.
  The reaction time is independent of the cycle time. Reactions to outputs are also carried out in defined time intervals.

Maximum number of sections

There can be up to a maximum of 1,600 sections per program.

Programming languages

Sections can be programmed using the IEC programming languages FBD (Function Block Diagram), LD (Ladder Diagram), SFC (Sequential Control), IL (Instruction List), or ST (Structured Text), or in the LL984 programming language (Ladder Logic), which resembles Modsoft. Only one of the stated programming languages is permitted to be used within a section.

Exchanging values

Values are exchanged within sections via links, variables, or direct addresses. Values are exchanged between different sections via variables or direct addresses.

Section execution order

The LL984 sections are the first to be executed. The LL984 section vertical sequence can be defined via the **Project →Configurator →Configure →Segment scheduler...** dialog box. Once the entire LL984 section has been processed, the IEC sections are then processed (FBD, SFC, LD, IL, ST). The execution order can be determined using either the **Project →Execution order...** or the Project browser *(see page 557)* dialog box.

Printing sections

Sections are divided into pages when printing out. The amount of information on these pages is dependent on the settings in the menu **File →Print**. Page division can be displayed using the menu option **View →Page breaks**.

Section variable

A Multi-element variable is automatically generated for each IEC section (FBD, SFC, LD, IL, and ST) and has the same name as the section.

This variable is SECT_CTRL data and has two elements:

- The "disable" BOOL data type element for disabling sections.
- The "hsbyState" BYTE data type element for displaying the Hot Standby status of sections.
  If the smallest bit of this element is set, the data from this section is transferred/received, see the *Hot Standby User's manual*. (This bit corresponds to the exclamation mark in the project browser.)

Disabling sections

The component "disable" can be used to enable/disable the section variable If the multi element address is not used or if the value 0 has been assigned to "disable", the corresponding section is executed. If "disable" is assigned the value "1", the corresponding section will not be executed. By using this variable, the execution of sections can be controlled according to events.

NOTE: If a disabled section is animated, the DISABLED status is displayed in the status bar.

⚠ CAUTION

Risk of unwanted process states.

Disabling a section does not mean that programmed outputs will be deactivated within the section if an output has already been set in a prior cycle, this status remains even after the section is disabled. The status of these outputs cannot be modified.

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

Disabling Interrupt Sections

A specific Interrupt section can be disabled using the ISECT_OFF block. It can be enabled again using the ISECT_ON block. The section names are provided by the SECT_CTRL control variable.

The I_LOCK block can disable all interrupt sections. They can be enabled again using the I_UNLOCK block.

NOTE: A possible interrupt on an interrupt section has no effect.

Lock section UNCONDITIONALLY (possibility 1)

The procedure for locking a section unconditionally is as follows:

Step	Action
1	Using **Online** →**Reference data editor** open the Reference data editor *(see page 595)*.
2	By double clicking on a line number, open the **Lookup variables** dialog box.
3	From the area **Data type** first choose the option **Structured** and then from this list **SECT_CTRL**. **Result:** The names of all sections are displayed.
4	Now select the names of the section to be locked.
5	Use the command button **Components...** to select the **ANY type components** dialog box.
6	Select the line **disable: BOOL** and confirm with **OK**.
7	If the following has not been performed yet: Create a connection between the PLC and the programming device and load your program onto the PLC.

Step	Action
8	Change the entry in the column **Value** to 1 (TRUE) to lock the section or 0 (FALSE) to enable the section.
9	Using **Online** →**Animation** activate the animation if it is inactive. **Result:** The section is disabled or enabled according to the value. **Note:** Locking a section does not mean that programmed outputs will be deactivated within the section if an output has already been set in a prior cycle, this status remains even after the section has been disabled. The status of these outputs cannot be modified.

⚠ **CAUTION**

Risk of unwanted process states.

The entry in the column **Value** remains even after the reference data editor has been closed (even if the entries are not saved), or in other words, the section remains disabled and must be explicitly re-enabled via the reference data editor (value = 0).

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

Lock section UNCONDITIONALLY (possibility 2)

The procedure for locking a section unconditionally is as follows:

Step	Action
1	Using **Project** →**Project browser** open the Project browser *(see page 557)*.
2	From **Online** →**Connect...** create a connection between the programming device and the PLC.
3	From **Online** →**Download...** (if the program is in **NOT EQUAL**mode) or **Online** →**Download changes** (if in **MODIFIED** mode) restore the consistency between the programming device and the PLC.
4	Select the section to be locked from the project browser.
5	Activate the context menu for sections using the right mouse button, and activate **Animate enable state**.
6	Change the enable status using the menu command **Switch enable state** from the context menu (right mouse button) of the selected section. **Note:** Sections may only be disabled or enabled via the Project browser, if they have not already been disabled/enabled via another Section *(see page 72)* or via the Reference data editor *(see page 70)*. **Result:** The section is locked. **Note:** Locking a section does not mean that programmed outputs will be deactivated within the section if an output has already been set in a prior cycle, this status remains even after the section has been disabled. The status of these outputs cannot be modified.

Locking a section CONDITIONALLY

The procedure for locking a section conditionally (program dependent) is as follows:

Step	Action
1	Create the logic according to the section to be locked, for example in an FBD section. When doing this, please note that the logic must carry a BOOL data "output" and that the section to be disabled will be disabled at logic "1". **Note:** The section containing a logic for disabling/enabling other sections should not be disabled.
2	By double clicking on your logic's "output", open the **Connect FFB** dialog box.
3	Use the command button **Lookup...** to open the **Lookup Variable** dialog box.
4	From the area **Data type** first choose the option **Structured** and then from this list **SECT_CTRL**. **Reaction:** The names of all sections are displayed.
5	By double clicking, now select the names of the section to be locked.
6	Select the line **disable: BOOL** and confirm with **OK**. **Result:** The multi-element variable from the section to be locked (Section name.disable) now creates the "output" of the logic.
7	From **Project** →**Execution order...** open the **Section Execution Order** dialog box.
8	Using the command buttons, ensure that the section containing the logic for locking is executed before the section to be locking is executed.
9	If the following has not been performed yet: Create a connection between the PLC and the programming device.
10	Download your program to the PLC. **Result:** When logic "1" is at the "Output" the section to be locked is not edited. **Note:** Locking a section does not mean that programmed outputs will be deactivated within the section if an output has already been set in a prior cycle, this status remains even after the section has been disabled. The status of these outputs cannot be modified.

Configuration data

Description

The PLC configuration is the interface between the program and the hardware.

The configuration data consists essentially of the component list and the entry in the address field of the program.

Loadables facilitate communication with the IEC programming language and the loading of further LL984-Instructions.

Creating a Project

4

Overview

This chapter describes the general procedure for the initial creation of a project. The most linear sequence possible is used here, in order to show a Concept-newcomer an easily manageable way of creating a project. Crosslinks between the Menu Commands are of course possible. As they gain experience, users will learn shortcuts and alternatives. For more detailed information, please see the relevant chapters in the user manual.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Overview	76
Step 1: Launching Concept	77
Step 2: Configuring the PLC	78
Step 2.1: Required Configuration	79
Step 2.2: Optional Configuration	81
Step 3: Creating the User Program	85
Step 4: Save	88
Step 5: Perform Memory Prediction	89
Step 6: Loading and Testing	90
Step 7: Optimize and Separate	95
Step 8: Documentation	97

Overview

Project Creation

The creation of a project has 8 main steps:

Step	Action
1	**Launching Concept** *(see page 77)* Launch Concept and start a new project.
2	**Configuring the PLC** *(see page 78)* Set the hardware configuration.
3	**Creating the user program** *(see page 85)* Create new sections and create your program.
4	**Save** *(see page 88)* Save your project
5	**Perform Memory Prediction** *(see page 89)* Check the PLC memory workload.
6	**Loading and testing the project** *(see page 90)* Create a link between the PC and the PLC. Load the project in the PLC and start it. Test the program with the Online Test Function. Now eliminate any mistakes in the program! Load the altered sections into the PLC.
7	**Optimize and Separate** *(see page 95)* It is now advisable to optimize the program storage capacity and to reload the optimized program into the PLC. After successfully loading, testing and (if necessary) optimizing, you may disconnect the PC from the PLC. The program will now run offline.
8	**Documenting** *(see page 97)* Create a complete set of documentation of your project.

Notes

NOTE: The steps "Configuring the PLC" and "Creating the User Program" can be performed in either order. This means that the PLC configuration can also be changed after the creation of the program.

NOTE: In order to prevent loss of data, you should save your program regularly.

Step 1: Launching Concept

Launching Concept

The procedure for launching Concept is as follows:

Step	Action
1	Double click on the Concept icon to launch Concept.
2	Select **File** →**New Project**.
3	You can specify a new project path or accept the standard project path with the project name `namenlos.prj`. **Result:**The new project is opened. **Note:** If you select the standard project path with the project name `namenlos.prj` , you can save this project with a name at a later time *Step 4: Save, page 88*. A saved project can be invoked with the **Open Project...**, or by using its project icon.

Note

NOTE: For additional steps please note the settings in the submenu **Options** → **Preferences**!

Resume

Now proceed with Step 2: Configuring the PLC *(see page 78)*.

Step 2: Configuring the PLC

What should be configured?

Using **Project →PLC configuration** configure the entire hardware configuration for your project.

Required Configuration

NOTE: The PLC type must first be set! All further configurations can then be executed independently of the processing sequence.

The following configurations are necessary for the configuration:

- *Specifying the type of PLC (minimum configuration), page 79*
- *Set memory partitions, page 79*
- *Install loadables, page 80*
- *Set I/O map, page 80*

Optional Configuration

The following configurations are to be used according to the project:

- *Set head setup, page 81*
- *Set Modbus communication, page 81*
- *Set Peer Cop communication, page 82*
- *Set data protection, page 82*
- *Various PLC settings, page 83*
- *ASCII messages (only for 984 LL), page 83*

Step 2.1: Required Configuration

Precondition

The PLC type must first be set! All further configurations can then be executed independently of the processing sequence.

Specifying the type of PLC (minimum configuration)

The procedure for specifying the type of PLC (minimum configuration) is as follows:

Step	Action
1	Select **Project** →**PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **PLC Selection** menu command from the list. **Response:** The **PLC selection** dialog is opened.
3	From the **PLC family** list select your PLC type.
4	Select your CPU from the **CPU/Executive** list.
5	From the **Runtime** list select the **Enable** status. **Response:** It is possible to program sections in IEC languages (FBD, LD, IL and ST). **Note:** In the **Runtime** list, the status **Not available**, **Disabled** or **Only 984** is displayed, then the selected CPU does not support any IEC programming languages. If in the list the status **Only IEC** is displayed, then the selected CPU exclusively supports IEC languages and these do not have to be explicitly enabled.
6	With simple tests and programs the configuration can now be exited and the procedure continued from *Step 3: Creating the User Program, page 85* or*Step 4: Save, page 88*.

Set memory partitions

The procedure for setting the memory partition is as follows:

Step	Action
1	Select **Project** →**PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **PLC memory partition** menu command from the list. **Response:** The **PLC memory partition** dialog is opened.
3	In the **Discretes** and **Words** ranges select the probable number of I/O flag bits and I/O words, to be required by the user program **Note:** The maximum address range, that must not be exceeded, can be read on the right-hand side of the dialog.

Install loadables

The procedure for installing the loadables is as follows:

Step	Action
1	Select **Project** →**PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **Loadables** menu command from the list box. **Response:** The **Loadables** dialog is opened.
3	Select the loadable in the **Available:** list. **Note:** Loadables are assigned in the *Loadables, page 114*section.
4	Select the **Install =>** command button. **Response:** The selected loadable is moved to the **Installed:** field.
5	Repeat the steps 3 and 4 until all the loadables required have been installed.

Set I/O map

The procedure for setting the I/O map is as follows:

Step	Action
1	Select **Project** →**PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **I/O map** menu command from the list. **Response:** The **I/O map** dialog is opened.
3	Select the **Supervision time** column and enter a time, within which a communication exchange must take place. If this time is exceeded, an error message appears.
4	Select the **Edit...** command button. **Response:** The dialog for entering modules is opened.
5	In the **Module** column, select the **...** command button. **Response:** The **I/O Module Selection** dialog is opened.
6	In the **Modules** column, select the module. **Response:** The module is displayed in the current slot.
7	Select the **Input start** and/or **Output start** columns and enter the first address of the occupied input and/or output reference range for the module.
8	Select the module and choose the **Params**command button. **Response:** If the module has a parameter dialog, you can define the parameter (e.g. disconnect behavior, data format, measuring range) here.

Resume

Now proceed with Step 3: Creating the user program *(see page 85)*.

Step 2.2: Optional Configuration

General Information

The following configurations do not need to be executed urgently, but they offer extended functions.

Set head setup

The procedure for specifying the remote I/O is as follows (this procedure is optional for minimum configuration):

Step	Action
1	Select **Project →PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **I/O map** menu command from the list. **Response:** The **I/O map** dialog is opened.
3	Select the **Head setup...** command button. **Response:** The **Head Setup** dialog is opened.
4	Enter the slots for the RIO or NOM modules. **Response:** Return to the **I/O map** dialog.
5	Select the head setup in the **Go To** list.
6	Select an empty line (last line) in the table, and select the **Insert**command button. **Response:** In the **Type** column another I/O station is entered.
7	Select the **Drop** column and enter the station number. **Note:** Only as many remote I/O stations can be configured as there are segments registered in the segment scheduler.
8	Select the head setup in the **Go To** list for the 2nd drop.
9	Next, carry out steps 3 to 6 of the *Set I/O map, page 80* procedure.

Set Modbus communication

To set the Modbus communication (Quantum slave, terminal, printer, etc.) proceed as follows:

Step	Action
1	Select **Project →PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **Modbus Port settings** menu command from the list. **Response:** The **Modbus port settings** dialog is opened.
3	Make the corresponding settings.

Set Peer Cop communication

If a Modbus Plus link exists, the Peer Cop functionality is able to transfer state RAM data globally or directly between several nodes on a local network. The procedure for setting the Peer Cop communication is as follows:

Step	Action
1	Select **Project →PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **Config. Extensions →Select Extensions** list. **Response:** The **Select extensions** dialog is opened.
3	Check the **Peer Cop** box. **Response:** Return to the **PLC configuration** window and the **Peer Cop** menu command is now available.
4	Select **Config. Extensions →Peer Cop**. **Response:** The **Peer Cop** dialog is opened.
5	In the **Go To** range select the local bus devices, and enter the slot.
6	Select in the **Global** range the **Receive...** and **Send...** command buttons to define the destination and source addresses of the transmission data and/or the address of the other bus devices.
7	Select in the **Specific** range the **Receive...** and **Send...** command buttons to define the destination and source addresses of the transmission data and/or the address of the other bus devices.

Set data protection

Address ranges of coils and holding registers can be protected from being overwritten by external signals. The procedure for setting the data protection is as follows:

Step	Action
1	Select **Project →PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **Config. Extensions →Configuration extensions**. **Response:** The **Configuration extensions** dialog is opened.
3	Check the **Data protection** box. **Response:** Return to the **PLC configuration** window and the **Data protection** menu command is now available.
4	Select **Config. Extensions →Data protection**. **Response:** The **Data protection** dialog is opened.
5	Select the range for the coils and holding registers. This range should contain write-protection.

Various PLC settings

Diverse internal PLC data can be evaluated, a watchdog timeout for the user program can be specified, the time windows for the communication (I/O time disk) parameterized and the multiple assignment of outputs authorized. The procedure for setting the PLC settings is as follows:

Step	Action
1	Select **Project** →**PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **Specials** menu command from the list. **Response:** The **Specials** dialog is opened.
3	Check the **Battery coil**, **Timer register** and **Time of Day** check boxes and enter an address in the corresponding text boxes.
4	Check the **Allow Duplicate Coils** check box and enter the address from which this should be allowed in the text box..
5	In the **Watchdog timeout (ms*10):** text box enter a numeric value between 2 and 255 (ms). This enables you to set an impulse watchdog for the user program. **Response:** As soon as the count pulses exceed the specified time, an error message appears.
6	In the **Online Editing Timeslice (ms):** text box enter a numeric value between 3 and 100 (ms). This enables you to define a time for executing the multi-cycle edit functions (paste, delete, find etc.)

ASCII messages (only for 984 LL)

To set the ASCII messages (only for 984LL), execute the following steps:

Step	Action
1	Select **Project** →**PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select from the list **ASCII** →**ASCII Setup**. **Response:** The **ASCII Setup** dialog is opened.
3	Enter the total messages, the size of the message width and the number of ASCII ports (from the I/O periphery) in the text boxes. **Response:** In the **PLC configuration** →**ASCII** window the **ASCII Port Settings** menu command is available.
4	Select from the list **ASCII** →**ASCII port settings**. **Response:** The **ASCII port settings** dialog is opened.
5	Make the corresponding settings. **Note:** ASCII messages can now be created under **Project** →**ASCII messages...** .

Resume

Now proceed with Step 3: Creating the user program *(see page 85)*.

Step 3: Creating the User Program

General

A user program is created in sections. Each section is programmable in one of the available languages and has a unique name in the project. Sections can be generated at any time during the programming.

Overview

The creation of a user program consists of 9 steps:

Step	Action
1	Generating a New Section *(see page 85)*
2	Declaring the Variables *(see page 86)*
3	Programming a Section *(see page 86)*
4	Analyzing Program/Section *(see page 86)*
5	Specifying the section execution sequence *(see page 87)*

Generating a New Section

The procedure for generating a new section is as follows:

Step	Action
1	In the main menu **File** call up the menu command **New section...** . **Result:** The dialog box **New program section** is opened.
2	Click on the programming language desired for this section.
3	In the text box **Section name** enter the unique name for this section.
4	Generate all the required sections in this way.

Declaring the Variables

A program consists of functions and Function Blocks (FFBs) or of instructions with the statement of variables (e.g. signals), addresses or literals. While direct addresses and literals can be used immediately, variables must be declared before they can be used in programming. The procedure for declaring variables is as follows:

Step	Action
1	In the main menu **Project** call the menu command **Variable declaration...** . **Result:** The dialog box **Variable declaration** is opened.
2	Enter the variable name, the associated data type, and if necessary the reference address, the initial value and a comment.
3	Confirm the entries with **OK**. **Note:** Further editing is also possible from a FFB connection or contact etc. by double-clicking -> **Var. Declaration...** . This starts the Variables editor.

Programming a Section

The procedure for programming a section is as follows:

Step	Action
1	Using **File** →**Open section** open the section to be programmed.
2	Create programs according to the rules of the individual programming languages: ● Function Block Diagram FBD *(see page 213)* ● Ladder Diagram LD (IEC) *(see page 239)* ● SFC (Sequential Control) *(see page 271)* ● Instruction list (IL) *(see page 321)* ● Structured text (ST) *(see page 393)* ● LL984 (Ladder Diagram (Modsoft)) *(see page 455)*

Analyzing Program/Section

Check a section or the entire program for syntax violations! The procedure for analyzing a program/section is as follows:

Step	Action
1	In the main menu **Project** call up the menu command **Analyze section** or **Analyze program**.
2	Remove the cause of the displayed or reported error. **Note:** Loading a section or program into the PLC is only possible after an error-free check. (The removal of the cause of warnings is not absolutely necessary. Checking the warnings is, however, sensible.)

Set execution order of sections

The sections are initially stored in the order of their creation and are executed after the program has started. In general this sequence must be adjusted project-specifically to suit the task setting. The procedure for specifying the section execution sequence is as follows:

Step	Action
1	To specify the section execution sequence there are two alternatives: ● In the main menu **Project** call the menu command **Execution order...** and using the command buttons **First**, **Last**, **Next**, **Previous** sequence the sections as required. ● In the main menu **Project** call up the menu command **Project browser** and sequence them as required by moving them around in the *Project Browser, page 557*.

Resume

Now proceed with Step 4: Saving *(see page 88)*.

Step 4: Save

General Information

General information about saving:

- If you exit a project without saving, you will be automatically asked if you want to save the project or not. If you answer yes to this question, this begins the same procedure described below.
- In order to prevent loss of data, projects should be saved regularly during long periods of configuration or programming sessions.

Saving a Project for the First Time

The procedure for saving a project for the first time is as follows:

Step	Action
1	In the **File** main menu invoke the **Save Project As...** menu command.
2	In the **File name** text box, enter the project name name.prj.
3	Select the desired drive and directory from the **Directory** list. Alternatively, it is possible to enter the whole path specification in the **File name** text box, e.g. **`c:\product1\reactor3.prj`** `(max. 28 characters + .prj)`. If these directories do not yet exist, they will be automatically created. **Note:** According to IEC 1131, a project includes all programs, data etc which belong to a PLC. If several projects (i.e. PLCs) belong to one system, then all projects should be stored in a common directory named after the system.
4	Click the **OK** command button. **Response:** The project has now been stored in the specified directory under the given name.

Supplementary Saving

The procedure for supplementary saving is as follows:

Step	Action
1	From the **File** main menu simply select the **Save** menu command.

Resume

Now proceed with Step 5: Executing memory prediction *(see page 89)*.

Step 5: Perform Memory Prediction

Check the PLC memory workload.

Perform an offline memory prediction of the configured PLC before downloading the program to the PLC. The table displayed in the **Project →Memory Prediction** dialog shows the use of individual memory ranges. An expected memory workload is then recognized.

NOTE: In some cases the memory prediction is not very accurate. A discrepancy between required memory in the PLC and the memory prediction under Concept may occur. The memory prediction always indicates more available memory than is actually available in the PLC.

This is due to the dynamic memory in the DFBs and Sections, which is difficult to calculate. Especially ST sections cause a great difference between the prediction and PLC. To be sure that there is sufficient memory available in the PLC, load a project into a PLC for examination. The simulator cannot be used because many projects have sufficient memory in the simulator but not in the PLC.

Resume

Now proceed with Step 6: Loading and testing the project *(see page 90)*.

Step 6: Loading and Testing

General Information

Loading and testing programs is only possible if

- either the 16-bit simulator Concept SIM is switched on or
- the Concept SIM 16-bit simulator is switched off and a PLC is attached with a Modbus Plus, Modbus, TCP/IP cable, or
- the Concept PLCSIM32 simulator is switched on.

NOTE: Testing using Concept SIM *(see page 746)* and Concept PLCSIM32 *(see page 748)* simulators is only possible with IEC user programs.

Overview

Loading and testing macros is divided into 9 main steps:

Step	Action
1	Loading the EXEC file into the PLC (see *Concept Installation Instructions*)
2	Connecting the PC and PLC *(see page 90)*
3	Loading and Starting the Program *(see page 91)*
4	Activating the Animation *(see page 91)*
5	Changing the Values of Literals *(see page 92)*
6	Changing the Values of Variables *(see page 93)*
7	Locating Errors *(see page 93)*
8	Downloading Changes *(see page 94)*
9	Starting and Stopping the PLC *(see page 94)*

Connecting the PC and PLC

The procedure for linking the PC and the PLC is as follows:

Step	Action
1	From the **Online** main menu invoke the **Connect...** menu command. **Response:** The **Link to PLC** dialog box opens.
2	Set the protocol type (Modbus, Modbus Plus, TCP/IP or Simulator) and the PLC node (when working in a network) with which you wish to communicate.
3	Under **Access right** select the **Change Configuration** option
4	Confirm the details with **OK**.

Loading and Starting the Program

The procedure for loading and launching the program is as follows:

Step	Action
1	From the **Online** main menu invoke the **Connect...** menu command. **Response:** The **Download Controller** dialog box will be opened in the PLC.
2	When loading the program for the first time, use the **All** command button.
3	Click the **Load** command button. **Response:** Various dialog boxes will be displayed.
4	Answer the question `Stop the program in PLC? Yes/No` with **Yes**. **Note:** This question only appears when a program is already running in the PLC.
5	Answer the question `Start a program in PLC? Yes/No` with **Yes**, if there are no errors. If warnings or errors are reported, these will be listed in the **Messages** window. Correct the warnings or errors at the specified point.

Activating the Animation

With the animation (online status report) it is possible to monitor the status of variables, steps, transitions etc within individual sections of the editor window. The procedure for activating the animation is as follows:

If…	Then…
To display binary values exclusively.	To display binary values exclusively, invoke the **Online** main menu and click on the **Animate booleans** menu command. **Response:** The valences of all booleans (variables, direct addresses, literals) are displayed in colour (0-Signal = red, 1-Signal = green).

If…	Then…
If you want to display the values of all variables.	To display the values of all variables invoke the **Editing** main menu option and select the **Select All** menu command (selects all items in the current section). Thereafter invoke from the **Online** main menu option the **Animate selection** menu command. **Response:** The valences of all values (variables, direct addresses, literals) are displayed in colour (red = 0-Signal, green = 1-Signal, yellow = either, for variables, immediate display of the value or, for multi-element-variables, displays the value by double-clicking on the variable).
If you want to enter monitoring fields in the text languages (IL and ST).	Use the **Selected Inspect** menu command to paste the text languages IL and ST into section monitoring fields. **Response:** The current value of the allocated variables is shown in these monitoring fields. With multi element variables, only the value of the first element is shown. This can be changed by double-clicking on the monitoring field of the **Numeric Inspect Settings** dialog box, which invokes the options available.

Changing the Values of Literals

The procedure for changing literals is as follows:

Step	Action
1	Activate the animation, as described in *Activating the Animation, page 91*.
2	Double-click on the literal to be changed.
3	Enter a new value and confirm with **OK**. **Response:** The new value will be sent to the PLC during the next logic scan.

Changing the Values of Variables

With the Reference data editor *(see page 595)* it is possible to show and set the values of variables (state, control, force). The procedure for changing variables is as follows:

Step	Action
1	From the main menu, select **Online** and then the **Reference data editor** menu command.
2	Enter the variables to be displayed in the dialog box marked **RDE Templates**.
3	To set the value highlight the **Disable** check box, and enter the desired value.
4	The RDE template can be saved under a unique name. To do this, invoke the **RDE** main menu option and select the **Save template as…** menu command. **Note:** Several RDE templates can be invoked at once. To do this, invoke the **RDE** main menu option and select the **Open template...** menu command.

Locating Errors

If errors occur during the processing of the program by the PLC, these will generally be reported on screen **Messages** and entered in an events list in log book form. The procedure for locating errors is as follows:

Step	Action
1	From the **Online** main menu invoke the **Event Viewer** menu command. **Response:** A window is opened, in which all errors are listed and described.
2	Select an error line and use the command button **Go to Error**. **Response:** This will go directly to the section in which the error occurred. The faulty object is highlighted.
3	Correct the program.
4	If your program now has the **UNEQUAL** status carry out the steps in Downloading and Starting the Program *(see page 91)* once again. If the program now has the **MODIFIED** status perform the steps in Downloading Changes *(see page 94)* once again.

Downloading Changes

If the project has the **MODIFIED** status after it has been altered, these changes can be loaded online into the PLC without stopping the program currently running. The procedure for downloading changes is as follows:

Step	Action
1	From the **Online** main menu access the **Download Changes...** menu command.
2	Click on **OK**. **Response:** The changes will be downloaded to the controller.

Starting and Stopping the PLC

The procedure for starting and stopping the PLC is as follows:

Step	Action
1	If the same project is running on the PC and PLC (**EQUAL**), then the PLC can be started or stopped with **Online** →**Online Control Panel...** .

Resume

Now proceed with Step 7: Optimize and Separate *(see page 95)*.

Step 7: Optimize and Separate

Optimizing Projects

At the end of the installation and/or after several runs of**Download Changes...** it is useful to perform an optimization, so that any gaps in the program data memory management are filled. After optimization the project is **UNEQUAL** on the PC and PLC and the program must be loaded into the PLC with **Download...** (**Warning:** Program must be stopped and restarted!). The procedure for optimizing projects is as follows:

Step	Action
1	Save the project with **File** →**Save Project**.
2	In the **File** main menu invoke the **Close project** menu command and take note of the dialog boxes which then appear.
3	In the **File** main menu invoke the **Optimize Project...** menu command and select the project to be optimized. Take note of the dialog boxes which subsequently appear.
4	Check the size of the program data memory in the **Online** main menu with the **Memory Statistics...** menu command.
5	The sizes can then be altered with **PLC configuration**.
6	Save the project with **File** →**Save Project**.
7	Reload the optimized program into the PLC using **Online** →**Download...** . To do this the program currently running must be stopped.
8	Start the newly loaded program using **Online** →**Online Control Panel**.

Separating the PC and Controller

After successfully testing the program in the PLC (with a connected process) the PC can be separated from the controller. The procedure for separating the PC and the controller is as follows:

Step	Action
1	Please take note of the program status in the footnote! To maintain consistency **EQUAL** must be there. ● if it reads**MODIFIED**, modifications must be loaded first *Downloading Changes, page 94*. ● If it reads**UNEQUAL** the program must be reloaded into the PLC *Loading and Starting the Program, page 91*.
2	From the **Online** main menu access the **Disconnect...** menu command. Take note of the information in the displayed dialog box.
3	The project can be closed after separation. In the **File** main menu invoke the **Close project...** menu command. Take note of the information in the dialog box, if displayed.

Resume

Now proceed with Step 8: Documenting *(see page 95)*.

Step 8: Documentation

General information

Each project should be fully documented. Changes and additions should also be documented (partial documentation).

Among other things documentation includes:
- Comments on the project (**Project** →**Properties**),
- Comments on each separate section (**File** →**Section properties**),
- Comments on variables,
- Comments on the functions applied, function modules and DFBs (command button **Comment** in the property dialog of each module),
- Comments on steps and transitions (command button **Comment** in the property dialog of each element),
- Comments in the form of freely placed text elements in the graphic programming languages (**Object** →**Text**),
- Comments on each line of commands in the textual programming languages
- Comments on user-specific data types,
- Comments on derived function modules (DFBs).

Printing the documentation

The procedure for printing documentation is as follows:

Step	Action
1	In the main menu call up **File** menu command **Print...** .
2	In dialog box **Documentation contents** select **Page layout** whether each page should have a uniform header and footer as well as printing a front page. The appearance of header, footer and front page is stored in the available ASCII files.
3	In the area**Contents** and in dialog box **Documentation contents**, select what is to be printed.
4	If **Variable list** has been selected, call up **Options** in order to select the variables which are to be printed.
5	When **Sections** has been selected, ● call up **Select** and specify the sections that are to be printed and ● also call up **Options**. In area **Graphics enlargement factor** also specify the appropriate size of the logic which is to be printed.
6	Activate command button **OK**. **Reaction:** All entries are saved.
7	Make sure that the page set-up of the sections is as desired. In the main menu call up **View**follow this with the successive menu commands **Overview** and **Pabe Break**.
8	Change the order of for example the FFBs in such a way, that there are as few transitions between adjoining pages as possible.
9	In the main menu call up **File** the menu command**Print...**again and activate command button **Print**. The printout is made with defined settings and the dialog box is closed.

PLC configuration

5

Overview

This section describes the single process for the hardware configuration.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
5.1	General information about hardware configuration	100
5.2	Configuration in OFFLINE and ONLINE mode	103
5.3	Unconditional Configuration	107
5.4	Optional configuration	122
5.5	Backplane Expander Config	133
5.6	Configuration of various network systems	137
5.7	Quantum Security Settings in the Configurator	151

5.1 General information about hardware configuration

Overview

This section contains general information about hardware configuration.

What's in this Section?

This section contains the following topics:

Topic	Page
General information	101
Proceed in the following way with the configuration	102

General information

At a Glance

The system configuration has far-reaching consequences as it influences the entire control work mode. It has to define all control-specific information as well as general information, allocate the necessary memory space and determine the input/output area. For the first configuration the user must enter several basic details for the PLC area, such as PLC type and memory.   Only valid configurations are authorized.

A configuration always refers to a Project, i.e. the menu command **PLC configuration**is only available when a project has been opened.

The configuration is available offline or online.

Proceed in the following way with the configuration

Introduction

In this section you are given a general overview on how to proceed with the configuration.

Use Configuration Menu

There are menu commands that absolutely must be carried out and are available in the **PLC Configuration** window. Grayed out menu commands are currently unavailable and can be enabled for extending the hardware-configuration in the **Config. Extensions** directory with the menu command **Select Extensions**.

Read in Module Set-up

The PLC module set-up is entered manually and can be compared with the connected hardware in ONLINE mode. After it has been read in, the modules missing in Concept are shown in the I/O map, and can be re-edited.

The I/O addressing must then be done for each module.

When doing this, please ensure the permitted references are used:

Modules	**References**
Analog input modules	3x references
Analog output modules	4x references
Digital input modules	3x or 1x references
Digital output modules	4x or 0x references
Expert modules - input	3x or 1x references
Expert modules - output	4x or 0x references

Downloading the Hardware Configuration

The hardware configuration of a project is saved and can be downloaded to the simulation program Concept-SIM, Concept-SIM32 or an automation installation. By doing this, the EQUAL status is established between the host computer and the PLC.

NOTE: The Concept-SIM must be deactivated for transfer of the configuration to a real PLC.

5.2 Configuration in OFFLINE and ONLINE mode

Overview

This section contains information for configuration in OFFLINE and ONLINE mode.

What's in this Section?

This section contains the following topics:

Topic	Page
General information	104
Available Functions in OFFLINE and ONLINE Modes	105

General information

At a Glance

In OFFLINE mode no link is created between programming device and PLC, and the configuration can be performed. In ONLINE mode there is a link between programming device and PLC, so that only one conditional configuration can take place.

Available Functions in OFFLINE and ONLINE Modes

Introduction

This section contains an overview of the available functions in OFFLINE and/or ONLINE mode. The possibilities in the ONLINE mode are different in their use of the simulator and the real PLC.

Configuration in OFFLINE Mode

In OFFLINE mode all menu commands are available for the hardware configuration in the **PLC Configuration** window. The submenus in the **Config. Extensions** directory can be enabled in the **Select Extensions** dialog to extend the configuration.

If the PLC is in ONLINE mode, you can switch to OFFLINE mode using the menu command **Online →Disconnect...**. In the footer of the editor window, the status-bar indicator NOT CONNECTED appears.

Configuration in ONLINE Mode and in the Active Simulator

A configuration is not possible in ONLINE mode with an active simulator or a Modbus Plus connection, i.e. no entries can occur. The available dialogs can only be invoked and read.

You can switch to ONLINE mode using the menu command **Online →Connect...** and establishing a connection between the host computer and the PLC.

Configuration in ONLINE Mode and in the Real PLC

Using the connection to a real PLC a configuration in ONLINE mode is possible, as long as the **Change Configuration** access level is activated.

It is not possible to configure or reconfigure a PLC while the PLC is in RUN mode. If a program is already running in the PLC, it must be stopped before reconfiguration can be implemented. Stop the PLC with **Online →Online Control Panel →Stop PLC**. After editing, the changes are automatically transferred to the hardware when the PLC is started up.

NOTE: When you delete an Expert module in ONLINE mode in the I/O map, the allocated loadable is also automatically deleted. If you wish to place this module back in the I/O map at a later time, it will be necessary to download again.

You can switch to ONLINE mode using the menu command **Online →Connect...** and establishing a connection between the host computer and PLC.

Effects of ONLINE Changes

If the following conditions are satisfied, all animated windows are automatically closed if a change is made in the I/O map (e.g. deleting or adding to a module)

Conditions:

- ONLINE mode
- animated section(s)
- Status between PLC and host computer is EQUAL
- Controller stopped
- Access level **Change Configuration** is activated.

5.3 Unconditional Configuration

Overview

This section contains a description of the configuration to be performed unconditionally and an overview of the presettings in the configuration menu.

What's in this Section?

This section contains the following topics:

Topic	Page
Precondition	108
PLC selection	109
CPU Selection for the PLC Type	110
PLC memory mapping	113
Loadables	114
Segment manager	117
I/O Map	119

Precondition

Introduction

Only when the CPU has been selected in the **PLC Selection** dialog will all the other menu commands become available in the **PLC Configuration** window.

The following dialogs are a minimum selection and MUST be edited as part of the hardware configuration.

- **PLC Selection**
- **PLC Memory Partition**
- **Loadables**
- **Segment Scheduler**
- **I/O Map**

The preferences can be adopted as long as they are compatible with the hardware being used.

PLC selection

Introduction

Select the PLC family (Quantum, Compact, Momentum or Atrium) and the CPU, as well as the memory size, according to use. All the available CPUs are listed in the list box.

Determine logic zone

The logic zone for the desired programming language (IEC or LL984) can be expanded to the corresponding PLC type with the PLC family selection.

The assignment and installation of the loadables is determined according to the following settings:

Selection	Meaning
Enable	Installation of the IEC loadables. A desired memory area for the IEC zone can be set up. The assignment and installation of the loadable pairing to the selected CPU is performed automatically in the **Loadables** dialog.
Disable	No installation of the IEC loadables. This will completely switch off the IEC zone and the entire logic zone will be made available for the LL984.
984 only/IEC only	Some Momentum CPUs can only be programmed in the IEC zone or only in the LL984 zone.

Determine total IEC memory

By defining the total IEC memory size and the global data, you also automatically determine the IEC-program memory size. On the basis of this size, the available memory space for the LL984 user program can also be determined.

NOTE: With global data it is the memory space of the unlocated variables.

NOTE: Total IEC memory = IEC program memory + global data

CPU Selection for the PLC Type

Introduction

When installing hardware (Concept EXECLoader), you are required to load various EXEC data files (*.BIN). This determines the firmware for various PLC types. The available PLC types, which can be operated by loading the EXEC data files with the corresponding CPUs, are shown in the following tables.

Loading Firmware for Quantum PLC Types

The following table shows the current EXEC versions, which are located on the Service Release CD and supplied with Concept.

Quantum PLC type:

140 CPU	**Q186Vxxx (IEC+LL984)**	**Q486Vxxx (IEC+LL984)**	**Q58Vxxxx (IEC+LL984)**	**Q5RVxxxx (IEC+LL984)**	**QIECVxxx (IEC only) ***	**IEC memory (kbyte)**
113 02	X (LL984 only)	-	-	-	-	
113 02S	-	-	-	-	X	max. 150
113 02X	X (LL984 only)	-	-	-	-	
113 03	X	-	-	-	-	max. 136
113 03S	-	-	-	-	X	max. 379
113 03X	X	-	-	-	-	max. 136
213 04	X	-	-	-	-	max. 305
213 04S	-	-	-	-	X	max. 610
213 04X	X	-	-	-	-	max. 305
424 0x	-	X	-	-	-	max. 465
424 0xX	-	X	-	-	-	max. 465
434 12	-	-	X	-	-	max. 890
534 14	-	-	X	-	-	max. 2550
434 12A (Redesigned CPU)	-	-	-	X	-	max. 890
534 14A/B (Redesigned CPU	-	-	-	X	-	max. 2550

NOTE: * After the QIECVxxx.BIN EXEC data file has been loaded, the EMUQ.EXE loadable must be loaded into Concept in the **Loadables** dialog.

Loading Firmware for Quantum LL984 Hot Standby Mode

The Quantum CPUs not ending in X or S can be used for the LL984 Hot Standby mode. A special EXEC file must be downloaded onto the CPU for this. The loadable for LL984 Hot Standby (CHS_208.DAT) is automatically installed by the system.

Loading Firmware for Quantum IEC Hot Standby Mode

The 140 CPU 434 12 and 140 CPU 534 14 CPUs can also be used for IEC Hot Standby. A special EXEC file must be downloaded onto the CPU for this. The loadables for IEC Hot Standby (IHSB196.EXE and CHS_208.DAT) are automatically installed by the system.

Loading Firmware for Quantum Equation Editor

The Quantum CPUs not ending in X or S can be used for the LL984 equation editor. A special EXEC file must be downloaded onto the CPU flash for this. This EXEC file is not part of the Concept delivery range but can be obtained over the Internet at www.schneiderautomation.com.

Loading Firmware for Momentum PLC Type

The following table shows the current EXEC versions, which are located on the Service Release CD and supplied with Concept.

Momentum PLC type (CPU 171 CBB 970 30):

171 CBB	**MPSV100.BIN (LL984 only)**	**MPSV100e.BIN (IEC only)**	**IEC memory (kbyte)**
970 30-984	X	-	
970 30-IEC	-	X	236

Momentum PLC type (CPU 171 CCC 7x0 x0):

171 CCC	**M1LLVxxx (LL984 only)**	**M1IVxxxE (IEC only)**	**IEC memory (kbyte)**
760 10-984	X	-	
760 10-IEC	-	X	220
780 10-984	X	-	
780 10-IEC	-	X	220

Momentum PLC type (CPU 171 CCC 9x0 x0):

171 CCC	M1EVxxx (LL984 only)	M1EVxxxE (IEC only)	IEC memory (kbyte)
960 20-984	X	-	
960 30-984	X	-	
960 30-IEC	-	X	236
980 20-984	X	-	
980 30-984	X	-	
980 30-IEC	-	X	236

Momentum PLC type (CPU 171 CCS 7x0 x0):

171 CCS	M1LLVxxx (LL984 only)	M1IVxxxE (IEC only)	IEC memory (kbyte)
700 10	X	-	
700/780 00	X	-	
760 00-984	X	-	
760 00-IEC	-	X	160

The stripped EXEC of the M1 supports up to a maximum of 44 I/O modules.

Loading Firmware for Compact PLC Types

The **CTSXxxxD.BIN** EXEC file must be downloaded onto the CPU flash for all Compact CPUs.

Loading Firmware for Atrium PLC Types

A special EXEC file (see table below) must be downloaded onto the CPU flash for each Atrium CPU.

180 CCO	EXEC file
121 01	AI3Vxxxx.BIN
241 01 241 11	AI5Vxxxx.BIN AI5Vxxxx.BIN

PLC memory mapping

At a Glance

For the creation of the program, sufficient address zones for the necessary number of input bits, output/flag bits, input words and output/flag words are to be entered.

An overview of the state RAM value is also given:

- Max. state RAM
- State RAM in use
- State RAM use

An unassociated value is shown with an error message, and can be automatically suited to the given value.

IEC Hot Standby data

After configuration of an IEC Hot Standby system, enter sufficient address zones for the required number of input words. The higher the number of IEC Hot Standby input words, the larger the transmit buffers for the IEC component. This means all the bigger the IEC application in use can be.

⚠ **CAUTION**

System cycle time influence!

The size of the configured state RAM in an IEC Hot Standby project has a significant effect on the system cycle time. As soon as a configured cycle ends, the next starts after the transfer of all state RAM data to the CHS module.

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

Loadables

Introduction

Loadables are loadable programs, which are only loaded into the PLC when required.

The various uses of loadables are described in the following sections.

NOTE: When you delete an Expert module in online mode in the I/O map the allocated loadable is also automatically deleted. If you wish to place this module back in the I/O map at a later time, it will be necessary to download again.

Downloading Loadables for the IEC Runtime System

The following loadables for the combined execution of IEC and LL984 programs (CPU 113 0x, CPU 213 0x or CPU 424 02) are available:

If...	Then...
you want to use CPUs with the mathematics processor for IEC programming,	install the loadable pairing @1S7196 and @2I7196.
you want to use CPUs without the mathematics processor for IEC programming,	install the loadable pairing @1SE196 and @2IE196.

Downloading Loadables for Expert Modules

The following loadables are available for Expert modules:

If...	Then...
you are configuring the 140 ESI 062 00 module with 32 bit runtime system and the 140-NOA-611-x0 module	install the loadable ASUP196. **Note:** The ULEX196 loadable is automatically installed. The ASUP 196 loadable is only installed automatically on 32-bit CPUs. On 16-bit CPUs with Stripped EXEC (QIECVxxx.BIN), the ASUP196 loadable must be installed afterwards.
you are configuring the 140 ESI 062 10 module,	install the loadable pairing NSUP + ESI. **Note:** These two loadables do not come with the Concept software package, but are supplied with the 140 ESI 062 10 module and must be unpacked at the time of installation (**Unpack...**).

Downloading Loadables for LL984

These are not included in the Concept delivery range. You can order these loadables via the "Automation Customer Service Bulletin Board (BBS)" (related topics README).

Downloading Loadables for Hot Standby

The following loadables for Hot Standby mode are available:

If...	Then
you are using the LL984 Hot Standby mode,	the loadable CHS_208 is automatically installed.
you are using the IEC Hot Standby mode,	the loadables IHSB196 and CHS_208 will be loaded automatically.

Downloading User Loadables

Loadables that are created by the user are called user loadables (*.EXE, *.DAT). They are located in the Concept directory DAT and using the **Unpack...** command button they can be inserted into the **Loadables** dialog at installation.

Downloading Loadables for IEC Support Only

The following loadables for IEC support only (CPU 113 xxS without mathematics processor) are available:

If...	Then
your application uses REAL arithmetic,	install the loadable EMUQ196. **Note:** The loadable is installed together with the EXEC-file QIECVxxx (installation in Concept EXECLoader).

Downloading Loadables for INTERBUS and IEC Support Only

The following loadables for IEC support are available:

If the CPU	Then
● 113 02S ● 113 03S ● 213 04S ● 534 14 ● 434 12 is configured,	install the loadable ASUP196. **Note:** The ULEX196 loadable is automatically installed. The ASUP 196 loadable is only installed automatically on 32-bit CPUs. On 16-bit CPUs with Stripped EXEC (QIECVxxx.BIN), the ASUP196 loadable must be installed afterwards.
113 03 is configured	install the loadable pairing @1SE196 + @2IE196. The ULEX196 loadable is automatically installed.
213 04 is configured,	install the loadable pairing @1S7196 + @2I7196. The ULEX196 loadable is automatically installed.

Downloading Loadables for INTERBUS and LL984 Support Only

The following loadables for LL984 support are available:

If the CPU	Then
● 113 02 ● 113 03 ● 213 04 is configured,	you can install the following loadables: ● ULEX196 ● @1S7196 + @2I7196 + ULEX196 **Note:** The ULEX196 loadable is automatically installed with this.
● 534 14 ● 434 12 is configured,	the loadables ASUP196 and ULEX196 will be loaded automatically.

Segment manager

At a Glance

If a remote I/O st. (Drop) is configured, the sequence and method of processing the LL984 section can be defined in the dialog box **Segment manager**.

When deleting (in the dialog box **I/O map**) a configured remote I/O st. (Drop), it is automatically deleted in the segment manager.

Mode of Functioning

Every I/O st. (Drop) is assigned a segment. It is therefore not permitted to enter fewer segments in the segment scheduler, than there are I/O st.s (Drops) configured in the I/O map. In the segment scheduler, the maximum segment numbers is by default set at 32.

The configurator checks the agreement between the two dialogs and classifies the I/O st.s (Drops) in the segment scheduler. A window informs you which I/O stations (Drops) have been inserted.

Altering the segment processing sequence

The sequence for segment processing can be altered manually, in that the segment number or I/O st. number can be edited in the corresponding line. For the local I/O st. (Drop), 1 is entered in the first line of the dialog box in the columns**In stat.** and **Out stat.** automatically.**1**

If no sequence was defined, the segments are processed in ascending order.

Sorting criteria for additional I/O st.s

Recently added I/O st.s (Drops) are classified in the segment manager according to the following criteria:

If…	Then…
A new I/O st. is added,	it is automatically classified behind the last available line.
All determined segments are already in use,	the last segment is reused for the input of the new I/O st. (Drop), i.e. a segment number can be repeated, as the stations are differentiated.

Available methods for segment processing

When setting the segment manager, the following methods of processing can be selected:

Processing type	**Meaning**
Continuous	Cyclic processing
Controlled	Manually controlled processing
WDT reset	Reset watchdog timer
End of logic	End of processing

NOTE: If subprograms are to be used in LL984, the last configured segment cannot be processed in the segment manager. The type of solution must unconditionally be **End of logic**.

Advanced settings in the segment manager

With the "Controlled" type of processing, only the reference numbers 0x and 1x are authorized, which determines when the logic for the corresponding section is processed.

The field **In. stat.** and **Out stat.** allow the input of corresponding I/O st. numbers, which must be configured. If a **0**is entered, no input/output is served by this segment number.

I/O Map

Introduction

In the I/O map, configure the I/O stations (drops) with the modules in use. Afterwards perform the I/O addressing and the parameterization of the configured modules.

Allocating Drops

Drop numbers can be allocated optionally except for the first one (from 2 to). The first drop number is automatically recognized as the local drop, and cannot be edited.

Configuring the Backplane Expander

The 140 XBE 100 00 module is necessary to expand the backplane. By doing this you can connect a second backplane, and gain 13 extra slots. The 140 XBE 100 00 module is mounted in both backplanes and, in addition, requires an independent power supply (power supply unit).

Expanded backplanes are configured in Concept in the first drop using slots 2-1 to 2-16.

A more detailed description about the configuration of expanded backplanes with the 140 XBE 100 00 module is given in the chapter *Backplane Expander Config, page 133*.

⚠ **CAUTION**

The slot assignment of the 140 XBE 100 00 is not shown in the configurator, so a double assignment is possible.

You should take note of the hardware slots of the module and the power supply, and should not occupy these slots with other modules in the I/O map.

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

NOTE: The flow of data via an expanded backplane is quicker than via the remote system.

Allocating the I/O Ranges

When allocating the I/O ranges the following references are allowed:

- 3x references for analog input modules
- 4x references for analog output modules
- 3x or 1x references for digital input modules
- 4x or 0x references for digital output modules
- 1x or 3x references for Expert modules (input)
- 0x or 4x references for Expert modules (output)

NOTE: The unique addressing is checked so that no addresses are occupied twice within the configuration.

Parameterization

Configured modules can be individually parameterized to determine the variable process conditioned settings.

Connection to other Network Systems

In addition to local and remote drops, links to other network systems can be established with configured coupling modules:

- Ethernet
- INTERBUS
- Profibus DP

See also the chapter entitled *Configuration of various network systems, page 137* and *Configuration examples, page 877*.

Read in Map

In the ONLINE mode of the stopped PLC, the hardware modules are listed in the I/O map and can be transferred as follows:

Step	Action
1	Open a project.
2	Open the **PLC Configuration** window.
3	Using the **PLC Type** menu command, open the **PLC Type** dialog and select the PLC type.
4	Connect the host computer to the PLC (**Online** →**Connect...**).
5	Open the **I/O Map** dialog (**PLC Configuration** →**I/O Map**).
6	Use the **Edit** command button to open the **Local Quantum I/O station** dialog.
7	Check the **Poll** check box. **Response:** The recognized modules are listed in the **Read** column in color.
8	Double click on the colored text boxes in the **Read** column. **Response:** The listed modules are transferred to the **Module** column.
9	Enter the address zone in the corresponding columns (**In.Ref.**, **In End**, **Out Ref.**, **Out End**).
10	After the hardware matching between the host computer and the PLC, the configuration can continue.

5.4 Optional configuration

Overview

This section contains the description of the optional configuration.

What's in this Section?

This section contains the following topics:

Topic	Page
Settings for ASCII Messages	123
Making Additional Functions Available in the Configurator	124
Data Exchange between Nodes on the Modbus Plus Network	125
How many words are really used when data is received (Peer Cop)	126
Protecting Data in the State RAM before Access	128
Parameterize interfaces	129
Special Options	131

Settings for ASCII Messages

Introduction

To create the ASCII messages, you are required first of all to set a mask, which contains the number of messages, the message area size and the ASCII ports. Once you have done that you can create the ASCII messages, which are then processed with the Ladder Logic programming language.

Precondition

ASCII messages are only possible in the Quantum family, and can only be processed with the LL984 processing language.

Procedure

To create the ASCII messages, you must first set the mask:

Step	Action
1	In the **PLC Configuration** →**ASCII** window, open the **ASCII Setup** dialog.
2	In the **Total Messages** text box specify a value from 1 to 999.
3	In the **Message Area Size** text box specify a value from 1 to 9999 bytes.
4	In the **ASCII Ports** text box specify an interface from 2 to 32.
5	Confirm your entries with the **OK** command button. **Response:** The settings are saved and the dialog is exited.
6	In the **Project** main menu open the **ASCII Message Editor** dialog (with the **ASCII Messages...** menu command).
7	Create the ASCII messages here, see also the description *ASCII Message Editor, page 611*.

Making Additional Functions Available in the Configurator

Introduction

Additional functions can be used for the configuration, if they have previously been enabled or set in the **Select Extensions** dialog.

Activating Advanced functions/Dialogs

By checking the check box or setting the Ethernet modules the corresponding menu commands are enabled and can be edited in the **PLC Configuration →ASCII** window.

The following functions/dialogs can be activated:

- Data protection
- Peer Cop
- Hot Standby
- Ethernet I/O-Scanner

NOTE: The available functions are dependent upon the configured CPU. Also see the online help "Select Extensions".

Specify Coupling Modules

Coupling modules must be configured in order to connect to other network systems. To do this, specify the number of modules in the corresponding list box, which are then available in the I/O map.

The following systems can be configured:

- TCP/IP Ethernet
- Symax-Ethernet
- MMS-Ethernet
- Profibus DP

NOTE: The maximum number of coupling modules depends upon the configured CPU. Also see the online help "Select Extensions".

Data Exchange between Nodes on the Modbus Plus Network

Introduction

With a Modbus Plus (MB+) connection you can configure a PLC using the Peer Cop functionality, so that data exchange with another PLC is possible. In such a case, the Peer Cop takes data from a reference area within a "source" PLC and places this via the Modbus Plus (MB+) network into a determined reference range of a "destination" PLC. This operation is performed in the same identical way for each token rotation.

Using the Peer Processor, input data from other nodes on the local network can be received by the user program. Likewise, output data from the user program can be transmitted to other nodes on the local network.

The Peer Cop has two variants for data exchange:

- global data exchange
- specific data exchange

Precondition

The **Peer Cop** menu command is only available if, in the **Select extensions** dialog the **Peer Cop** check box is checked.

Global Data Exchange

With global data exchange, the data sent from the current "source" PLC is received by all "destination" PLC devices in the Modbus Plus (MB+) network. Up to 64 destination devices can be reached in this way, which can each receive the data in 8 destination addresses of the State RAM.

See also section "*How many words are really used when data is received (Peer Cop), page 126*".

Specific Data Exchange

With specific data exchange, data is sent from a selected "source" PLC to a selected "destination" PLC in the Modbus Plus (MB+) network. To do this, enter the respective addresses for the data exchange in a table at the corresponding source and destination nodes (1-64).

The address must correspond to the MB+ node address on the back of the respective module. This address setting can be altered and must be specified before mapping. (See also hardware description)

Select the node to be read or written according to the hardware configuration.

How many words are really used when data is received (Peer Cop)

Introduction

The number of words used may not exceed 500. To avoid this a simple formula can be used, how many words are used on receipt.

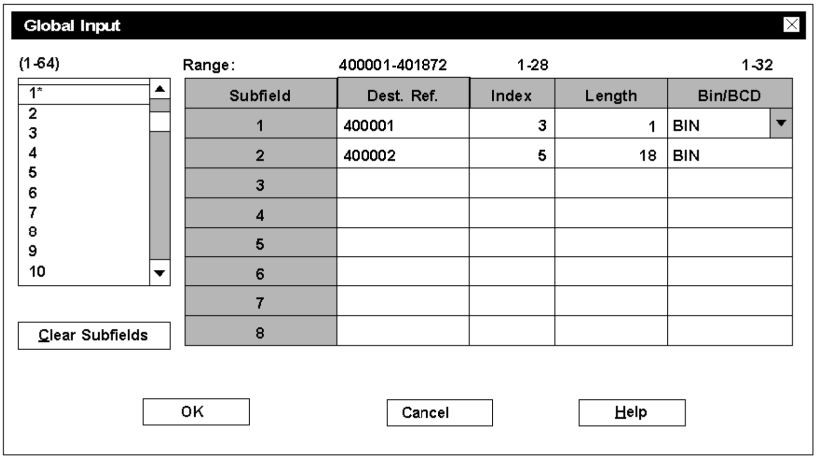
Formula

The formula, to find the number of words used is as follows:

Length + (index – 1) = number of words

Example

The Peer Cop dialog **Global Input** has the following entry:

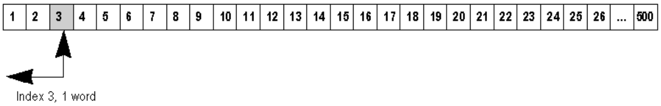
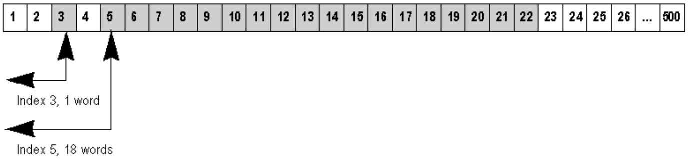


Global Input

Range: 400001-401872 1-28 1-32

Subfield	Dest. Ref.	Index	Length	Bin/BCD
1	400001	3	1	BIN
2	400002	5	18	BIN
3				
4				
5				
6				
7				
8				

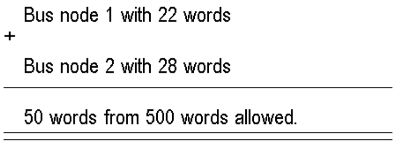
The following process takes place:

Step	Action
1.	Bus node 1 sends 1 word to the subfield start reference 400001, starting at index 3.
2.	At index 3 (word 3) the receipt of the data begins. (The preceding words are also counted.) Word 1 - 500  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 ... 500 Index 3, 1 word
3.	In total 3 words are required by subfield 1. Formula: 1 + (3 - 1) = 3
4.	Bus node 1 sends 18 words to the subfield start reference 400002, starting at index 5.
5.	At index 5 (word 5) the receipt of the data begins. (The preceding words are also counted.) Word 1 - 500  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 ... 500 Index 3, 1 word Index 5, 18 words
6.	In total 22 words are required by subfield 2. Formula: 18 + (5 - 1) = 22

NOTE: Only the largest number of words used per bus node by be taken into account. In the example 22 words from a maximum of 500 permitted words are used.

For more bus nodes the largest number of words used per bus node must be added.

For example:



Bus node 1 with 22 words

+

Bus node 2 with 28 words

50 words from 500 words allowed.

Protecting Data in the State RAM before Access

Introduction

Output address ranges (coils and registers) can be protected by specifying the address from which writing is possible in the **Data Protection** dialog. All addresses before this are write-protected.

Precondition

The **Data Protection** menu command is only available if, in the **Select Extensions** dialog, the **Data Protection** check box is checked.

Entering Access Protection

This access protection operates in connection with "normal" data access, which happens externally via a Modbus or Modbus Plus interface. Access from the host computer out is in any case permitted and bypasses this protection mechanism.

Parameterize interfaces

At a Glance

Depending on their use in Concept, the following interfaces must be parameterized:

- ASCII interface
- Modbus interface

Parameterize ASCII interface

For an ASCII message transmission, the serial communication parameters for the port interfaces can be specified in the **ASCII port settings** dialog box.

NOTE: The **ASCII port settings** dialog box is only available when the number of ASCII ports has been specified beforehand in the dialog box **ASCII set up**.

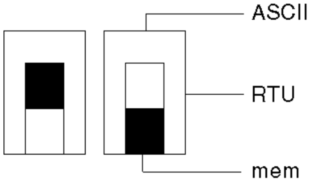
Parameterize Modbus interface

For a Modbus coupling, in the dialog box **Modbus port settings** the serial communication parameters of the port interface can be entered on the programming device, on a CPU and the NOM assemblies (Network Option Module).

⚠ CAUTION
Do not make any online changes since this will cause all Editors to close!
The Modbus port settings should not be altered in Online mode, or else all Editors are automatically closed.
Failure to follow these instructions can result in injury or equipment damage.

NOTE: The settings for a Modbus coupling in Concept only have an effect if the switch on the front of the assembly is at the lowest position (mem).

Switch position on the NOM



NOTE: If the left-hand switch is in the upper position and right-hand switch is set to mem then, as of firmware version 2.20, bridge mode is deactivated. This means that the network connection between Modbus and Modbus Plus is locked.

Interface parameterization with network connections between Modbus and Modbus Plus

A network connection between Modbus and Modbus Plus nodes can be made in the dialog box **Modbus port settings** by checking the check box **Bridge mode**.

NOTE: The settings are only effective if the switch on the front of the assembly is in the middle position (RTU).

Special Options

Introduction

In the **Specials** dialog you can configure special options:

- Battery coil
- Timer register:
- Time stamp for MMI applications (TOD)
- Allow duplicate coils
- Watchdog-Timeout (ms)
- Time slice for online changes (ms)

Battery coil

You can specify an address of a coil, which shows the status of the battery. This assignment is used for battery monitoring. In this way, the weak battery can be replaced early to avoid a loss of data.

Timer Register:

The content of the time register is incremented every 10 ms and has a free value between 0000 and FFFF hex.

Time for MMI applications (Date/Time)

This time stamp is only intended for a MMI application. Eight registers are reserved for setting the clock.

The TOD input (Time of Day) is in the American format:

4xxxx	Control register	
	Discrete 1 (MSB) Discrete 2 Discrete 3 Discrete 4	1 = set clock values 1 = read clock values 1 = preset discrete 1 = error discrete
4xxxx+1	Day of week (1 - 7)	
4xxxx+2	Month (1 - 12)	
4xxxx+3	Day (1 - 31)	
4xxxx+4	Year (00 - 99)	
4xxxx+5	Hours (0 - 23)	
4xxxx+6	Minutes (0 - 59)	
4xxxx+7	Seconds (0 - 59)	

Allow Duplicate Coils

You can assign several outputs to a coil. To do this, check the check box, and specify the first address to which several outputs can be allocated in the **First Coil Address:** text box.

NOTE: This function is unavailable with the Momentum PLC family.

Watchdog Timeout (ms*10)

You can set a pulse supervision for the user program by entering a numerical value of between 2 and 255 (ms). As soon as there are no count pulses within the specified time, an error message will appear.

Time Slice for Online Changes (ms)

You can set a time supervision for the communication between the nodes by entering a numerical value between 3 and 30 (ms). As soon as there is no communication within the specified time, an error message will appear.

5.5 Backplane Expander Config

Introduction

This chapter describes the function and configuration of the backplane expander.

What's in this Section?

This section contains the following topics:

Topic	Page
Generals to Backplane Expander	134
Edit I/O Map	135
Error handling	136

Generals to Backplane Expander

Introduction

The Quantum backplane expander provides a single backplane expansion to a local drop or a RIO drop through the 140 XBE 100 00 module.

Function description

The module connects two Quantum backplanes (primary and secondary) through a custom cable and support all data communication between the backplanes. Each backplane requires a 140XBE10000 module that occupy a single slot and requires its own power supply.

Procedure at an Error

The backplane expander is designed in the way that if it is not installed or improperly connected, it will not effect the functionality of the primary rack. Only the backplane expander installed and connected properly, the both racks are then able to communicate and controlled by prime CPU or RIO drop controller.

Edit I/O Map

Requirements

Currently only Quantum controllers support backplane expander. Primary rack contains the CPU or RIO drop controller and is allowed to config all type of additional modules up to the physical slot address limitation. All I/O modules can be also added to the secondary rack. However, option modules, such as NOMs, NOEs and CHSs must reside in the primary rack.

To place a module in proper rack, it is necessary to add an extra attribute in the I/O module database to specify that the module is available only for the primary or secondary or both.

Configuration in I/O Map

Exist Quantum local drop or RIO drop only support one rack up to sixteen slots. With backplane expander, it is extended as if the drop support two racks, and each has sixteen slots. By clicking at the button **...** on **Module** column, all modules available to the rack clicked (primary or secondary) will show in the module selection dialog that can be selected and assigned to the current slot.

Each rack requires a 140 XBE 100 00 module to make backplane expander work properly.

NOTE: The 140 XBE 100 00 module does not have a personality code and therefore can not be recognized by the Concept.

The module will just look like an unfilled slot in the Concept I/O map. If any module is configured in the secondary rack, it is user’s responsibility to ensure there is one slot in each rack that is reserved for 140 XBE 100 00 module and all hardware are connected properly.

Error handling

Introduction

The validate processes for the primary rack will be applied to the secondary rack too, such as duplicate reference, missing input or output reference, etc. Besides existing regular validation, traffic cop will do some special check for the backplane expander.

No reserved slot for 140 XBE 1000 00

If any module is found in the secondary rack and there is no empty slot left in either of racks when user trying to exit the rack editor dialog, an error message will be displayed: "There must be one empty slot reserved for 140 XBE 100 00 module in each rack to make backplane expander work." The rack editor dialog will then not be closed.

Special module in secondary rack

To prevent any special module (such as, NOE, CHS, etc) being added to the secondary rack, rack editor dialog do not allow to cut/copy these head modules. It will also check module personalities before user try to do any paste operation. If some unsupported module for the secondary rack is found, an error message will be displayed: "The buffer contains some module that can not reside in the secondary rack." The paste operation will be aborted.

5.6 Configuration of various network systems

Overview

This section contains the description of the configuration of various network systems.

What's in this Section?

This section contains the following topics:

Topic	Page
Configure INTERBUS system	138
Configure Profibus DP System	139
Configure Ethernet	141
RTU extension	143
Ethernet I/O Scanner	144
How to use the Ethernet / I/O Scanner	149

Configure INTERBUS system

At a Glance

The configuration of the INTERBUS system can take place within the PLC families of Quantum and Atrium.

INTERBUS configuration with Quantum

With the Quantum family the coupling of a remote bus takes place in a Quantum I/O station (Drop). To do this, the INTERBUS Master NOA 611 00 must be configured and parameterized in the CMD tool (Configuration Monitoring and Diagnostic Tool).

See also Configuration example 4 *(see page 910)*.

INTERBUS configuration with Atrium

With the Atrium family, the coupling of the remote bus takes place via the master assembly 180 CCO 121 01, 180 CCO 241 01 or 180 CCO 241 11 in this way, the INTERBUS Master CRP 660 0x is automatically inserted into the local I/O station (Drop). The INTERBUS I/O station (Drop) nodes are configured in the CMD tool (Configuration Monitoring and Diagnostic tool), saved as a *.SVC data file and imported to Concept. After the import into the I/O map the configuration can be changed afterwards in Concept.

See also Configuration example 9 *(see page 955)*.

Configure Profibus DP System

Introduction

The configuration of the Profibus DP system can take place within the PLC families of Quantum and Atrium.

Profibus DP Configuration with Quantum

With the Quantum family the connection to the Profibus DP system takes place in a Quantum drop. To do this, you must first of all set the number of bus controllers (CRP 811 00) used in the **Select Extensions** dialog. The modules then appear in the list box of the **I/O Module Selection** dialog and can be inserted into the I/O map.

The configuration of the Profibus DP node is created in the SyCon configuration tool, saved as a *.CNF file and transferred directly to Concept. However, the configuration (*.CNF) can be imported to Concept at a later time.

⚠ CAUTION

PROFIBUS DP ADDRESSES MAY BE OVERWRITTEN

When working with Profibus DP configuration make sure that the addresses of two 8 bit E/A modules without gap to the following 16 bit limit is only permitted when both 8 bit modules belong to the same Profibus DP master. If you do not adhere to this guideline, the input bits of one module (e.g. Profibus DP Master A) may be overwritten by the other module (e.g. Profibus DP Master B).

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

Importing the Profibus DP Configuration

To import the configuration (*.CNF) to Concept, proceed as follows:

Step	Action
1	In the **PLC Configuration** window, open the **I/O Map** dialog.
2	Select the drop and use the **Edit**dialog **Local Quantum I/O Drop**.
3	Double click on the in the **Module**column. **Reaction:** The **I/O Modules Selection** dialog is opened.
4	In the **I/O Adapter** column, select the CRP-811-00 module, and press the **OK** command button. **Reaction:** The CRP-811-00 will be inserted in the I/O map.
5	In the **Local Quantum I/O Drop** dialog, select the line of the mapped bus controller (CRP-811-00) and press the **Params** command button. **Reaction:** The **CRP-811-00 (Profibus DP)** dialog will open.
6	Using the **Import** open the **Select Import File** window.
7	To import, specify the path of the CNF file, and press the **OK** command button. **Reaction:** The Profibus DP configuration is entered in the Concept I/O map. **Note:** After the Profibus DP nodes are entered into Concept, the reference ranges for all modules and diagnostic data must be edited later.

Configuration Example

An example of configuration is given in Example 11 *(see page 925)*.

Configure Ethernet

Introduction

An Ethernet bus system can be configured within the following PLC families:

- Quantum
- Atrium
- Momentum

Precondition

In order to connect to the Ethernet bus system, a PCI network card must be available in the host computer. Afterwards the Ethernet interface needs to be parameterized and the drivers that are provided on CD need to be installed (*Configure Ethernet, page 975*).

After the Ethernet module has been slotted into the central backplane, the internet address, subnet mask, gateway and frame type can be allocated by the network administrator.

Configuration with Quantum

The procedure for Ethernet configuration in Concept is as follows:

Step	Action
1	In the **PLC Configuration** window, open the **Select Extensions** dialog.
2	Enter the number of Ethernet modules (NOE) in the text boxes. **Response:** The modules then appear in the list box in the **I/O Module Selection** dialog and can be inserted into the I/O map.
3	In the **PLC Configuration** window, open the **Ethernet I/O Scanner**dialog, in which you enter the information from the network administrator (Internet address, subnet mask, gateway, frame type).
4	In the **Online** main menu, open the **Connect to PLC** dialog (menu command **Connect...**).
5	In the **Protocol Type** list box, select the option **TCP/IP**, and in the **IP address or DNS Hostname** text box, enter the address of the TCP/IP card.
6	After programming, in the **Online** main menu, open the **Load into PLC** dialog (menu command **Load...**), and click on the **Load** command button. **Response:** A message appears, asking whether you would like to start the PLC.
7	Before you confirm the message with the **Yes** command button, the display "link" must appear on the Ethernet module.

Error Action

After configuration, only start the PLC once the display "link" has appeared on the Ethernet module. If this is not the case, withdraw the Ethernet module from the central backplane and then slot it in again. If the display "link" is still not shown, there must be a serious error.

Available Ethernet Modules

The maximum number of NOE modules is dependent upon the configured CPU (select in the **PLC Selection** dialog):

CPUs	Number of NOE modules
113 02/S/X	0 - 2
113 03/S/X	0 - 2
213 04/S/X	0 - 2
424 0x/X	0 - 6
434 12	0 - 6
534 14	0 - 6

Configuration with Momentum

The configuration of the Ethernet bus system with Momentum is described in the section *Momentum Example - Ethernet Bus System, page 974*.

RTU extension

Requirements

To make the RTU menu command available you have to choose a Compact CPU with LL984 programming language in the **PLC Selection** dialog.

CTS-/RTS-Delay

In this dialog you can set time delay for CTS or RTS independently for Comm port 1 of your Compact PLC. This feature allows modem communications with radios that require longer time frames. The delay time range is 0 ... 500 ms using 10 ms units. Enter the time delays your require.

Secured Data Area (SDA)

This feature allows you to configure an area in RAM that is secured from being overwritten. Secured Data Area (SDA) is a block of the Compact PLCs RAM that is set aside as 6x data space. The SDA can only be written to by specific functions that require secured data storage. General purpose Modbus commands, builtins, can not write to the SDA. Modbus Read (function 20) is able to read from the SDA, Modbus Write (function 21) is not able to write to the SDA. The SDA size range is 0 ... 128 K words using only 1 K word blocks. Enter the size your require.

Refer to the applicable user manual for the specific function for the required SDA size. For example, for Gas Flow, refer to the "Starling Associates Gas Flow Loadable Function Block" User Guide (890 USE 137 00).

PLC Login Password Protection

For the description of password protection, refer to section *Set PLC Password, page 660*.

Ethernet I/O Scanner

Introduction

This function is for the following Quantum modules available:
- 140-NOE-211-x0
- 140-NOE-251-x0
- 140-NOE-771-xx

This function is for the following Momentum modules available:
- 171-CBB-970-30
- 171-CCC-960-20
- 171-CCC-980-20
- 171-CCC-980-30
- 171-CCC-960-30

Ethernet address and I/O scanning parameters can be modified using the **Ethernet / I/O Scanner** dialog box. From the **PLC Configuration** window, select **Ethernet / I/O Scanner**. This menu option will only be available if you have selected an M1 Processor Adapter with an Ethernet port or have Quantum TCP/IP Ethternet modules (NOE) as specified above.

This section describes how to configure the Ethernet port, including IP address, other address parameters and I/O scanning.

Ethernet Configuration Options

The Ethernet / I/O Scanner screen offers three options for configuring the Ethernet port on an M1 Processor Adapter:

Configuration options	Meaning
Specify IP Address	This is the default option. It allows you to type the IP address, gateway and subnet mask in the text boxes in the upper righthand corner of the screen.
Use Bootp Server	Click this radio button if you want the address parameters to be assigned by a Bootp server. If you select this option, the address parameter text boxes in the upper righthand corner of the screen will be grayed out. They will not display the actual address parameters.
Disable Ethernet	Click this radio button if you want to disable the Ethernet port. Disabling the port will reduce the scan time for the Processor Adapter.

Setting Ethernet Address Parameters

If you choose to specify the IP address, you should complete all four text boxes in the upper righthand corner of the dialog box:

Parameters	Meaning
Internet Address	Type a valid IP address in the **Internet Address** text box (for example: 1.0.0.1).

⚠ CAUTION

Potential for duplicate addresses

Obtain a valid IP address from your system administrator to avoid duplication.

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

Gateway	Consult your system administrator to determine the appropriate gateway. Type it in the **Gateway** text box.
Subnet Mask	Consult your system administrator to obtain the appropriate subnet mask. Type it in the **Subnet Mask** text box (for example: 255.255.255.0).
Frame Type	For NOE there is an additional Frame Type field. Your two possible choices are ETHERNET II or IEEE 802.3

NOTE: Changing the subnet mask or the frame type and downloading the application via the NOE hinder the I/O scanner to run after a PLC start. Disconnecting Concept then leads to run the I/O scanner.

Configuring I/O

Once the Ethernet port address parameters have been set, you may assign parameters for I/O scanning.

The text box **Master Module (Slot)** contains the Module type that you have configured for Ethernet communications. In the case of the Momentum Ethernet controller the slot will always be number 1, and the configured module type is displayed in the variable dialog field. If you are configuring a NOE in a standard rack the slot number assigned in the I/O Map will be displayed along with the module type. Until the I/O Map is conmpleted this test field will indicate "Unassigned". In instances where more than one NOE is configured the I/O Scan parameters reflect the unit currently in the dialog box from which you can select the additional unit by activating the Pulldown list.

The text field **Health Block (1x/3x)** is only available by using the 140-NOE-771-xx. The health timeout is used for setting the health bit. If the response arrives before the end of the HealthTimeout period, the health bit is set; otherwise it is cleared. If the Health Timeout is zero, the health bit is set to true once communications are established, and it is never cleared.

NOTE: The configuration of the health block, refer to the user guide Quantum NOE 771 xx Ethernet Modules, model no. 840 USE 116 00.

The text box **Diagnostic Block (64 words, 3x or 4x registers)** is only available by using the Momentum Ethernet (M1E) and allows you to define the starting register of a number of bits which are used for diagnostic. The block can be specified in either 3x or 4x registers. For more information, refer to the user guide Quantum NOE 771 xx Ethernet Modules, model no. 840 USE 116 00.

The **Device Control Block** is only available when using NOE 771-01(11) with the Firmware Rev. 3.5 and higher.

Enable this check box, to enable/disable the I/O scanner entry.

Each entry in the I/O scanner can be enabled or disabled by setting the related bit in the control block.

- Bit=1, than I/O scanning stops, corresponding Health bit = 0 (socket is closed)
- Bit=0, than I/O scanning starts, corresponding Health bit = 1.

For more information, please refer to Quantum NOE 771 xx, Ethernet Modules, User Guide in Chapter I/O-Scanner Enable/Disable.

I/O Scanner Configuration table:

Column	Description
Slave IP Address	Type the IP address of the slave module in this column (for example: 128.7.32.54). This address will be stored in a pulldown menu, so that you may use it in another row by clicking on the down arrow and selecting it.
Unit ID	If the slave module is an I/O device attached to the specified slave module, use the Unit ID column to indicate the device number. The Unit ID is used with the Modbus Plus to Ethernet bridge to route to Modbus Plus networks.
Health Timeout	Use this column to specify the length of time in ms to try the transaction before timing out. Valid values are 0 ... 50 000 ms (1 min). To avoid timing out, specify 0.
Rep Rate	Use this column to specify how often in ms to repeat the transaction. Valid values are 0 ... 50 000 ms (1 min). **NOTE:** For legacy NOEs the repetition rate must be 0 or a multiple of 16 ms. Legacy NOEs are the NOE 771 00 versions and NOE 771 01/NOE 771 11 versions lower than rev. 4.3.

⚠ WARNING

Unpredictable operation

The repetition rate values of NOEs with firmware rev. 4.3 and higher must be 0 ms or a multiple of a step size between 5 ms (minimum) and 200 ms (maximum). The minimum cyclic repetition rate that is allowed is 5 ms.

If you configure more than one slave and want to use different repetition rates, then you must ensure that the repetition rates have a common step size.

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

Examples: If one repetition rate in the I/O scanner is set to 6 ms, all other repetition rates values must be a multiple of 6 (i.e. 6 ms, 12 ms, 18 ms, 24 ms, etc.)

- Repetition Rates 24 ms, 30 ms, 36 ms, 42 ms are also valid becaue the step size is 6 ms
- Repetition Rates 0 ms, 35 ms, 42 ms, 70 ms, 14 ms are valid because the step size is 7 ms
- Repetition Rates 24 ms, 35 ms, 19 ms are not valid because there is no common step size
- Repetition Rates 20 ms, 100 ms, 300 ms are not valid because the max. limit has been exceeded
- Repetition Rates 0 ms, 3 ms, 30 ms are not valid because the min. limit has been exceeded

I/O Scanner Configuration table continued:

Column	Description
Read Ref Master	Use the read function to read data from the slave to the master. This column specifies the first address to be read (for example: 400001).
Read Ref Slave	Use the read function to transfer data from the slave to the master. This column specifies the first address of up to 125 to read to (for example: 400050).
Read Length	Use the read function to read data from the slave to the master. This column specifies the number of registers to read (for example: 20).
Write Ref Master	Use the write function to write data from the master to the slave. This column specifies the first address to write (for example: 400100).
Write Ref Slave	Use the write function to write data from the master to the slave. This column specifies the first address of up to 100 to write to (for example: 400040).
Write Length	Use the write function to write data from the master to the slave. This column specifies the number of registers to write (for example: 40).
Description	You can type a brief description (up to 32 characters) of the transaction in this column.

NOTE: You may include read and write commands on the same line.

How to use

For more information about how to use the Ethernet / I/O Scanner dialog see section *How to use the Ethernet / I/O Scanner, page 149*.

How to use the Ethernet / I/O Scanner

Introduction

This section describes how to complete your Ethernet I/O configuration using the **Copy**, **Cut**, **Paste**, **Delete** and **Fill Down** buttons.

Copy and Paste

To save time when typing similar read and write commands, you may copy and paste entire rows within your configuration:

Step	Action
1	Select the row you want to copy by clicking on the row number at the far left.
2	Click the **Copy** button above the I/O configuration list.
3	Select the row where you would like to paste the data (by clicking on the row number at the far left).
4	Click the **Paste** button.

Cut and Paste

To move a row within the configuration list, follow the direction:

Step	Action
1	Select the row you want to move by clicking on the row number at the far left.
2	Click the **Cut** button above the I/O configuration list.
3	Select the row where you would like to paste the data (by clicking on the row number at the far left).
4	Click the **Paste** button. **Note:** Multiple rows may be cut/copy and pasted. The number of rows actually pasted is limited by the number of rows selected. For example if you copy 10 rows to the clipboard, then select an area of 6 rows to past, only the first six rows of clipboard data is pasted.

Delete

To delete a row within the configuration list, follow the direction:

Step	Action
1	Select the row you want to delete by clicking on the row number at the far left.
2	Click the **Delete** button above the I/O configuration list. **Note:** Multiple rows may be deleted.

Fill down

To copy part of any row to the next row or to a series of adjoining rows, use the **Fill Down** button, following the steps in the table

Step	Action
1	Use your mouse to select the data you would like to copy and the cells you would like to copy it to. **Note:** You must select one contiguous block of cells, with the data to be copied in the first row. You cannot select two separate blocks.
2	Click the **Fill down** Button. **Result:** The data from the first row is copied to the selected cells in the defined block.

NOE Ethernet modules

In this dialog the NOE Ethernet modules 140 NOE 211 x0,140 NOE 251 x0 and 140 NOE 771 10 are parameterized (in the **Ethernet Configuration** area).

In this dialog the NOE Ethernet module 140 NOE 771 00 is parameterized and addressed (in the **I/O Scanner Configuration** area).

For the followings modules you receive an function description:

- 140 NOE 211 x0
- 140 NOE 251 x0
- 140 NOE 771 xx

Momentum Ethernet modules

In this dialog the Momentum Ethernet modules are addressed (in the **I/O Scanner Configuration** area).

For the followings modules you receive an function description:

- 171 CBB 970 30 IEC
- 171 CBB 970 30 984
- 171 CCC 980 30 IEC
- 171 CCC 980 30 984
- 171 CCC 980 20 984
- 171 CCC 960 30 IEC
- 171 CCC 960 30 984
- 171 CCC 960 20 984

5.7 Quantum Security Settings in the Configurator

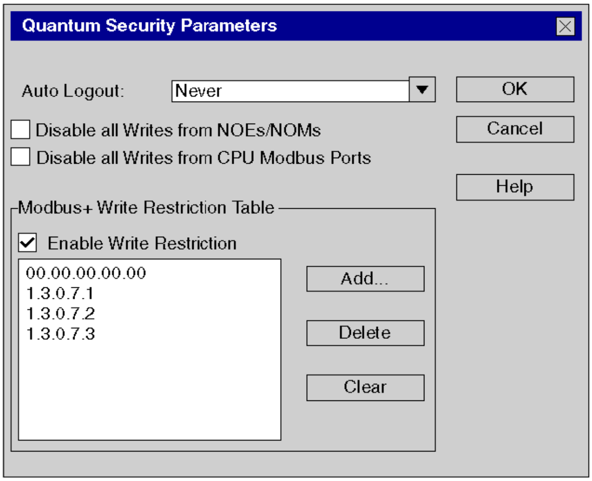
Quantum Security Parameters

Introduction

Various security parameters can be defined in the configuration of the Quantum CPUs 140 434 12A and 140 534 14A/B which are indicated in the log file *.LOG. This guarantees secure process documentation which includes the logging with the automatic logout, write access of NOEs/NOMs on the PLC as well as limited participants (max. 12) for network write access.

The definition of the security parameters can be found in dialog **Configuration** → **Security Expansion**.

Dialog **Quantum Security Parameters**:



Auto Logout: Never

□ Disable all Writes from NOEs/NOMs

□ Disable all Writes from CPU Modbus Ports

Modbus+ Write Restriction Table

☑ Enable Write Restriction

00.00.00.00.00
1.3.0.7.1
1.3.0.7.2
1.3.0.7.3

Requirements

The security parameters are only available if the following conditions have been met:

- Supervisor Rights (see Concept under **Help** →**Info...** →**Current User:**)
- only with CPUs 140 CPU 434 12A and 140 CPU 534 14A/B

Automatic Logout

Auto. logout is only available for Quantum CPU 434 12A and 534 14 A/B.

The automatic logout procedure logs a user out as soon as a predefined time limit (max. 90 minutes) is reached with no activity on the connection. This could be a lack of read or write activity from the programming device to the PLC for example.

The **Never** setting disables this function, i.e. automatic logout cannot occur.

NOTE: Automatic logout does not function if:

- the programming device (Concept) is connected to the PLC not via the local ModbusPlus Port of the CPU, but via a NOE/NOM module
  and at the same time
- another device is connected to the same NOE/NOM module with read access to the PLC.

Disable All Writes from NOEs/NOMs

By disabling all write accesses of

- 140 NBE 210 00 (ID-Code 0x0406)
- 140 NBE 250 00 (ID-Code 0x0407)
- 140 NOE 211 00 (ID-Code 0x0404)
- 140 NOE 251 00 (ID-Code 0x0405)
- 140 NOE 311 00 (ID-Code 0x0408)
- 140 NOE 351 00 (ID-Code 0x0409)
- 140 NOE 511 00 (ID-Code 0x040A)
- 140 NOE 551 00 (ID-Code 0x040B)
- 140 NOE 771 00 (ID-Code 0x040D)
- 140 NOE 771 01 (ID-Code 0x0422)
- 140 NOE 771 10 (ID-Code 0x040E)
- 140 NOE 771 11 (ID-Code 0x0423)
- 140 NOM 211 00 (ID-Code 0x010C)
- 140 NOM 212 00 (ID-Code 0x010C)
- 140 NOM 252 00 (ID-Code 0x010C)
- 140 NWM 100 00 (ID-Code 0x0420)

to the PLC, all write instructions are ignored by the CPU and responded to with an error message.

NOTE: MSTR read operations are not executed if the check box **Disable All Writes from NOEs/NOMs** is checked. (this also means the error state of the MSTR block shows no error!)

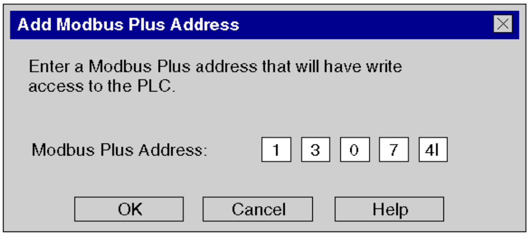
Disable all Writes from CPU Modbus Ports

To disable writes from the Quantum CPU Modbus connections, check the **Disable all Writes from CPUs from Modbus Ports** check box.

Limited Write Access on the Modbus Plus Network

A restricted number of participants that have access to the PLC can be configured for the Modbus Plus network. A maximum of 12 participants are allowed, the participant address of the programming device is automatically entered in the participant list and cannot be deleted.

Dialog **Add Modbus Plus Address** (press **Add...**)



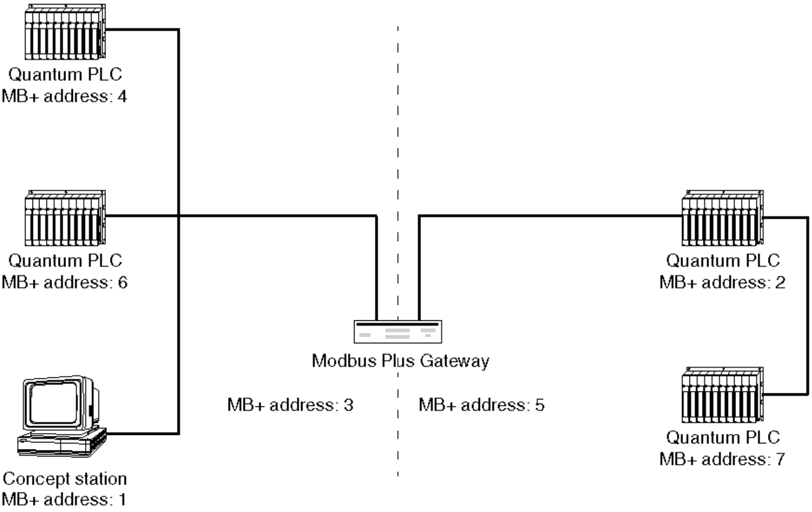
Add Modbus Plus Address

Enter a Modbus Plus address that will have write access to the PLC.

Modbus Plus Address: 1 3 0 7 4l

Examples of Modbus Plus paths

Modbus Plus network:



The address must be entered from the point of view of the receiving PLC to the sender, and thus begins at the first gateway or the next PLC. This depends on whether the sender and the receiver are located in the same Modbus Plus segment (no bridges/gateways) or whether the sender and the receiver are located in different segments (separated by one or more bridges/gateways).

Example 1:

Concept (MB+ Address 1) writes to PLC 6. There are no bridges or gateways between the two participants. Thus, the address entered appears as follows: 1 or 1.0.0.0.0

Example 2:

PLC 2 (MB+ Address 2) writes to PLC 6. A gateway (MB+ Address 3) is located between the two participants. Thus, the address entered appears as follows: 3.2.0.0.0

NOTE: Only the first Modbus Plus address can be recognized by the PLC. Thus, as soon as this first address is a bridge or a gateway, all devices in the network behind the bridge or gateway have write access to the PLC. Thus, in our example PLC 7 could also write to PLC 6 (Address: 3.7.0.0.0).

Main structure of PLC Memory and optimization of memory

6

Overview

This Chapter describes the main structure of the PLC Memory and the optimization of the memory with the different PLC families.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	Main structure of the PLC Memory	156
6.2	General Information on Memory Optimization	157
6.3	Memory Optimization for Quantum CPU X13 0X and 424 02	162
6.4	Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B)	176
6.5	Memory optimization for Compact CPUs	187
6.6	Memory optimization for Momentum CPUs	197
6.7	Memory optimization for Atrium CPUs	204

6.1 Main structure of the PLC Memory

General structure of the PLC Memory

At a Glance

In principle, the memory of a PLC consists of three parts:

- the memory for the Exec file,
- the state RAM and
- the program memory.

Memory for the EXEC file

The EXEC file contains the operating system and one or two runtime systems (IEC and/or LL984) for operating the user programs.

State RAM

The state RAM can be divided into different zones:

- the used 0x, 1x, 3x and 4x references,
- a reserve for further 0x, 1x, 3x and 4x references,
- possibly an extended memory zone for 6x references.

Program Memory

The program memory can be divided into different zones:

- the I/O map etc.,
- a reserve for extensions,
- the ASCII messages (if used), the Peer Cop configuration (if used), the Ethernet configuration (if used) etc.,
- a reserve for extensions,
- the IEC loadables (if required),
- the Global Data, consisting of the Unlocated Variables,
- the IEC program memory with the program codes, EFB-Codes and program data (section data and DFB instance data),
- possibly the ULEX loadable for INTERBUS or other loadables,
- the LL984 program memory.

6.2 General Information on Memory Optimization

Overview

This Section contains general information on memory optimization.

What's in this Section?

This section contains the following topics:

Topic	Page
Possibilities for Memory Optimization	158
PLC-Independent	159

Possibilities for Memory Optimization

Description

The possibilities for memory optimization are partly dependent on the PLC family and CPU used:

- *PLC-Independent, page 159*
- *Memory Optimization for Quantum CPU X13 0X and 424 02, page 162*
- *Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B), page 176*
- *Memory optimization for Compact CPUs, page 187*
- *Memory optimization for Momentum CPUs, page 197*
- *Memory optimization for Atrium CPUs, page 204*

PLC-Independent

Introduction

There are 3 PLC-independent possibilities for memory optimization:

- *Optimize State RAM for 0x and 1x References, page 159*
- *Only Download Required Loadables, page 160*
- *Optimize Expansion Size, page 161*

Optimize State RAM for 0x and 1x References

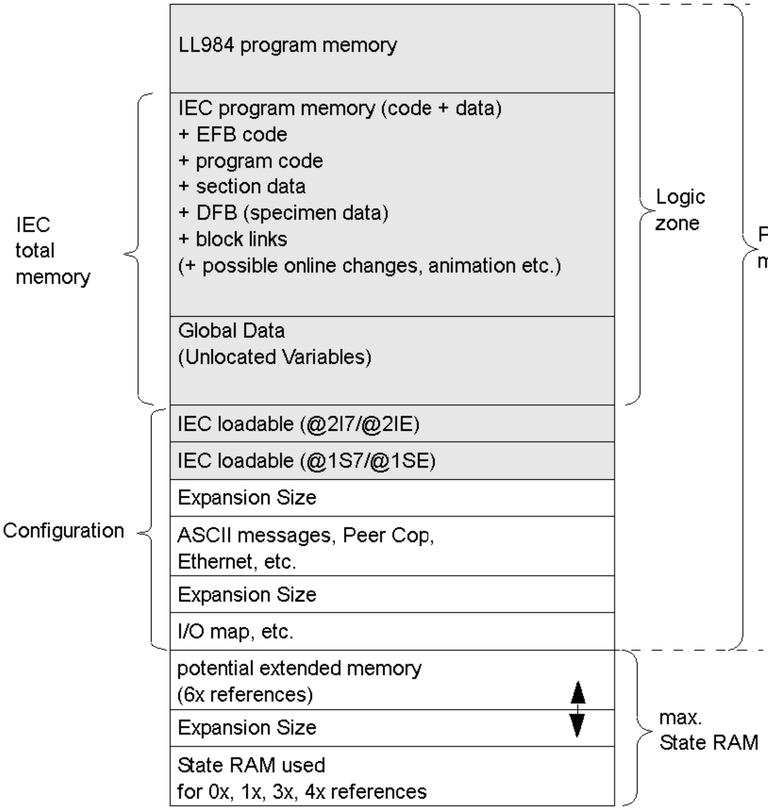
The state RAM contains the current values of the 0x, 1x, 3x and 4x references.

Even if the state RAM zone is outside the program memory zone, the size of the state RAM for 0x and 1x references influences the size of the program memory. Therefore, do not select a state RAM zone that is too large. In theory, the procedure only needs as many 0x and 1x references as the hardware requires. However, you will require a somewhat larger number of references if the I/O map is to be extended. It is advisable to be generous with the number of references during the creation phase of the user program when frequent changes are still being made. At the end of the programming phase, the number of these references can be reduced in order to create more space for the user program.

The settings for the 0x-, 1x-references can be found in **Project →PLC Configurator →PLC Memory Partition**.

In this dialog box, there is an overview of the size of the occupied state RAM zone and the percentage of the maximum state RAM that this represents.

Optimize state RAM for 0x, 1x, 3x and 4x references:



Only Download Required Loadables

All the installed loadables are downloaded into the program memory zone and occupy space. Therefore, only install those loadables which you really need (related topics *Loadables, page 114*).

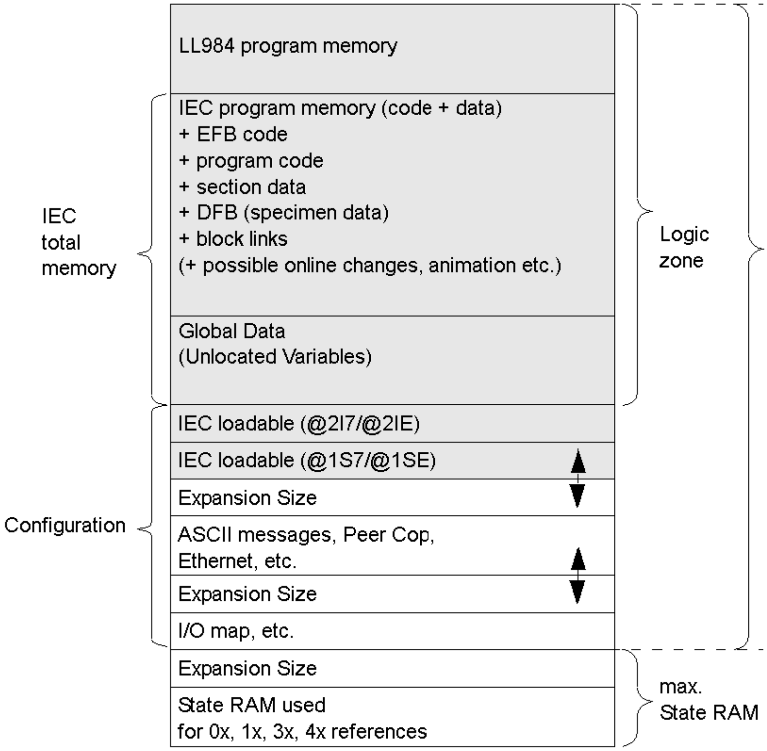
The memory space occupied by the installed loadables is displayed in the **Loadables** dialog box under **Used Bytes** (**Project** →**PLC configurator**). This information is calculated from the size of the loadable files and from the memory size assigned to the loadables.

Optimize Expansion Size

Each time, there is the possibility to reserve memory space for later expansion in the mapping zone (I/O map) and in the configuration expansion zone (Peer Cop). This memory space is necessary if e.g. the I/O map or the Peer Cop settings should be changed online. It is advisable to overestimate the reserves during the installation phase of the user program, that is, when modifications are often being made. At the end of the programming phase the reserves may be reduced again, to provide more space for the user program.

The settings for the mapping reserves are found in **Project** →**PLC Configurator** → **I/O Map** →**Expansion Size**. The settings for the Peer Cop reserves can be found in **Project** →**PLC Configurator** →**Config. Extensions** →**Select Extensions** →**Peer Cop** →**Expansion Size**.

Optimize Expansion Size



6.3 Memory Optimization for Quantum CPU X13 0X and 424 02

Overview

This Section describes the memory optimization for the Quantum CPUs CPU X13 0X and CPU 424 02.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information on Memory Optimization for Quantum CPU X13 0X and 424 02	163
Selecting Optimal EXEC File	165
Using the Extended Memory (State RAM for 6x references)	169
Harmonizing the IEC Zone and LL984 Zone	171
Harmonizing the Zones for Global Data and IEC Program Memory	173

General Information on Memory Optimization for Quantum CPU X13 0X and 424 02

Logic Memory

The program memory zone, in which the user program is located, is called the logic zone. This zone therefore determines the maximum size of your user program.

The current size of the logic zone is displayed under **Project →PLC Configuration** in the configurations overview in the **PLC** zone. The entry for the memory size is given in Nodes for LL984 (1 node equals 11 bytes) and in kilobytes for IEC.

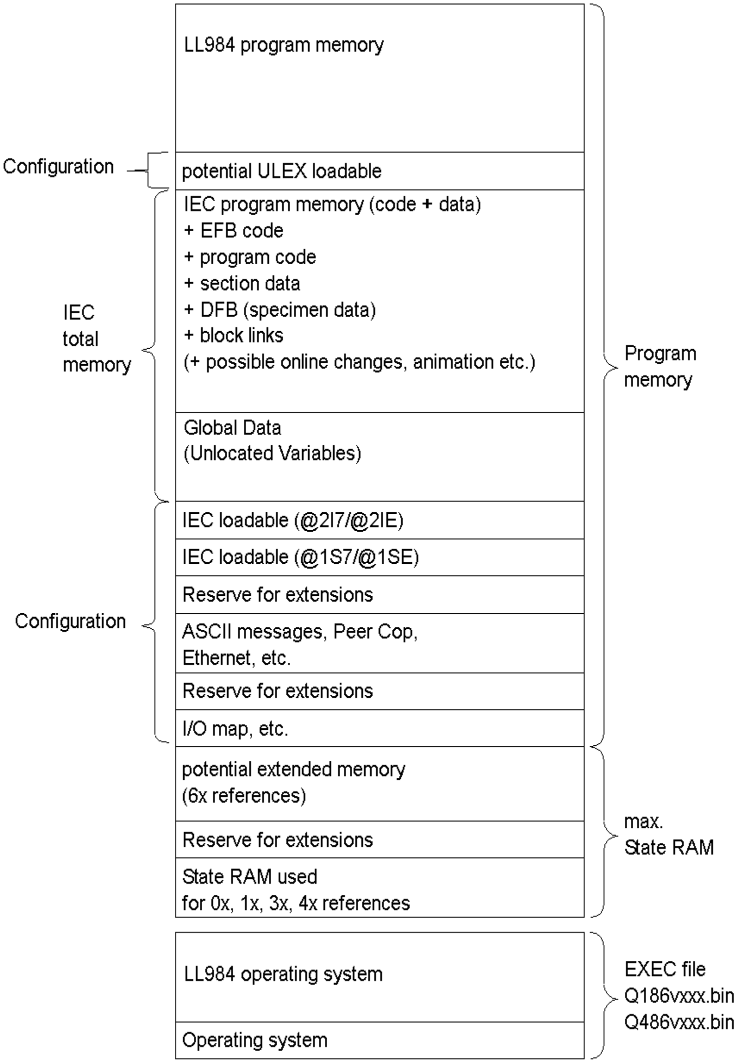
Optimizing the Logic Memory

You have various possibilities for optimising the logic memory to suit your requirements:

- *Selecting Optimal EXEC File, page 165*
- *Using the Extended Memory (State RAM for 6x references), page 169*
- *Harmonizing the IEC Zone and LL984 Zone, page 171*
- *Harmonizing the IEC Zone and LL984 Zone, page 171*

NOTE: Also note the PLC-independent possibilities for memory optimization *(see page 157)*.

Structure of the CPU X13 0X memory (simplified representation):



Selecting Optimal EXEC File

Introduction

The simplest and most basic option is to download the optimal EXEC file for your requirements onto the PLC (see also *Installation Instructions*).

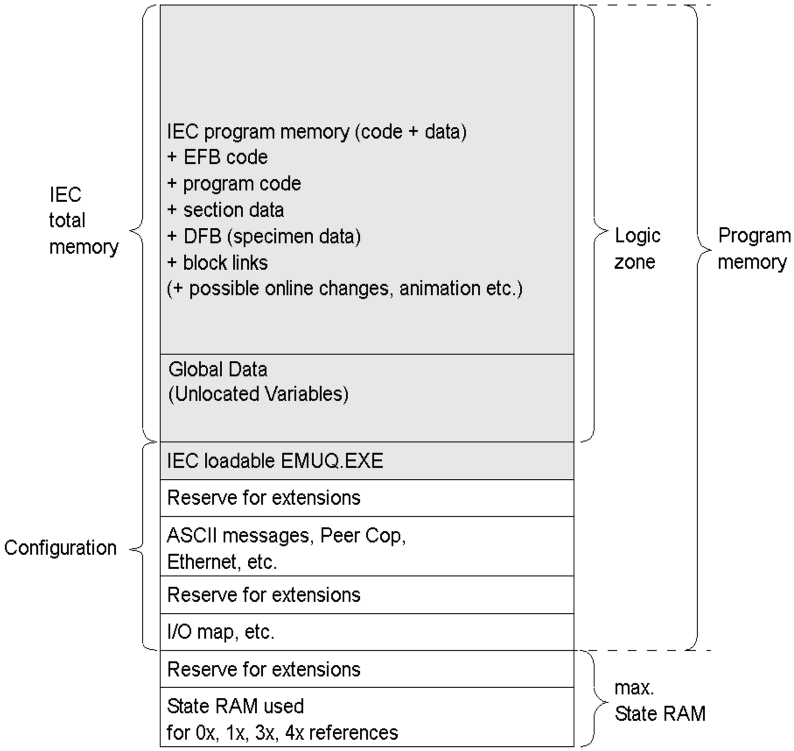
Depending on which EXEC file you select, zones will be reserved in the program memory of the PLC for IEC and/or LL984 programs. Therefore, if you install a 'combined EXEC file' and then only use one of the two language types in the user program, the program memory will not be used optimally.

Therefore, decide which languages you want to use:
- *Exclusive Use of IEC, page 165*
- *Exclusive Use of LL984, page 166*
- *Joint Use of IEC and LL984, page 167*

Exclusive Use of IEC

If you want to use IEC exclusively, download the EXEC file "QIEC_xxx.bin" (not available for CPU 424 02). Since this EXEC file does not contain an operating system, you have to download the IEC runtime system onto the PLC in the form of a loadable (EMUQ.exe) (related topics *Loadables, page 114*). The loadable is downloaded into the program memory zone and takes up memory space.

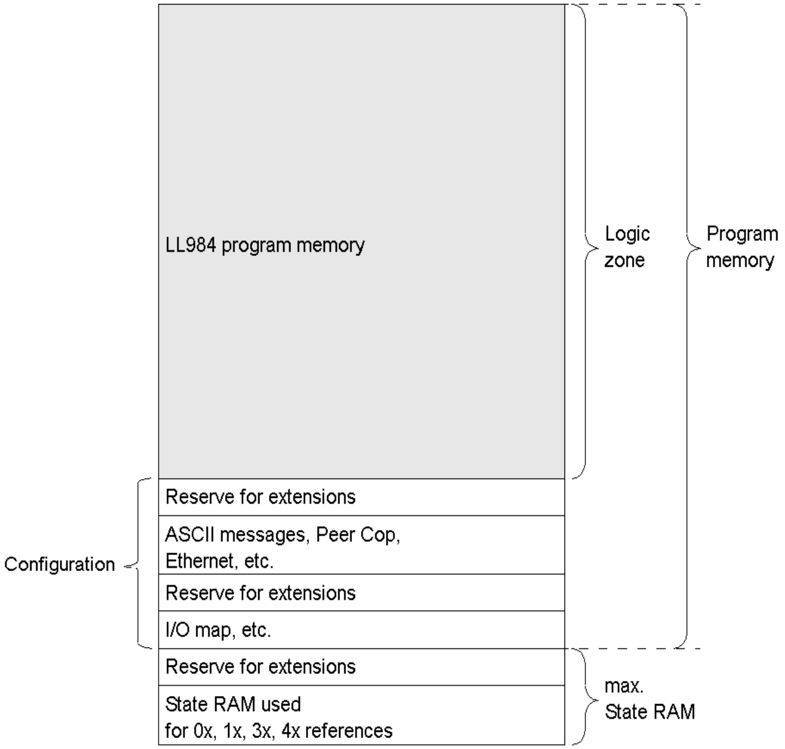
Structure of the CPU X13 0X memory with exclusive use of IEC:



Exclusive Use of LL984

If you want to use LL984 exclusively, download the EXEC file "Q186Vxxx.bin" for a CPU X13 0X and the EXEC file "Q486Vxxx.bin" for a CPU 424 02.

Structure of the CPU X13 0X memory with exclusive use of LL984:

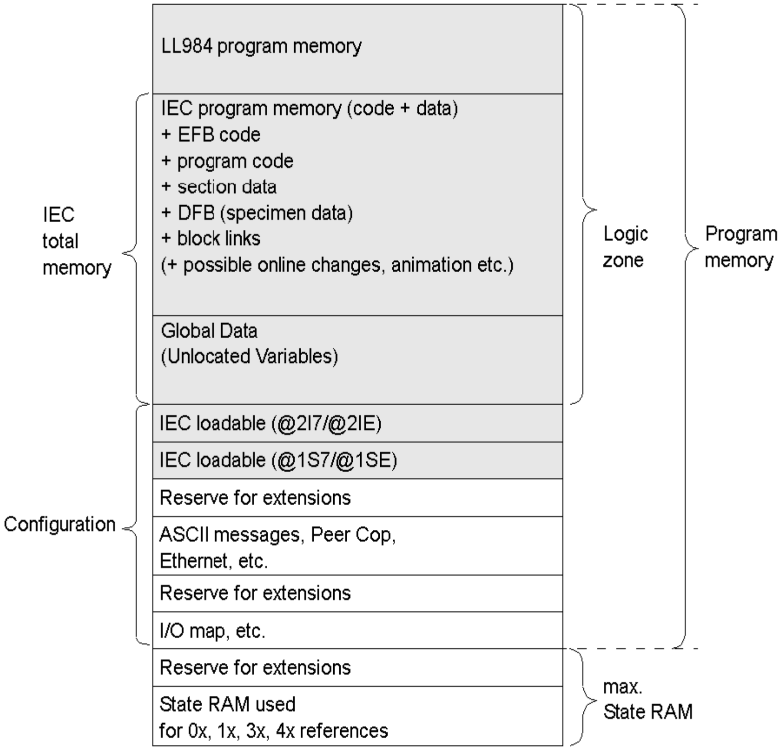


Joint Use of IEC and LL984

If joint use of IEC and LL984 is required, download the EXEC file "Q186Vxxx.bin" for a CPU X13 0X and the EXEC file zone "Q486Vxxx.bin" for a CPU 424 02. Since these EXEC files only contain the LL984 operating system, you have to download the IEC operating system onto the PLC in the form of loadables (@2I7/@2IE or @1S7/@1SE) (see also *Loadables, page 114*). Both loadables will be downloaded into the program memory zone and occupy memory space.

NOTE: Joint use of IEC and LL984 is not possible with the CPU 113 02 because its memory is too small for this application.

Structure of the CPU X13 0X memory with joint use of IEC and LL984:



Using the Extended Memory (State RAM for 6x references)

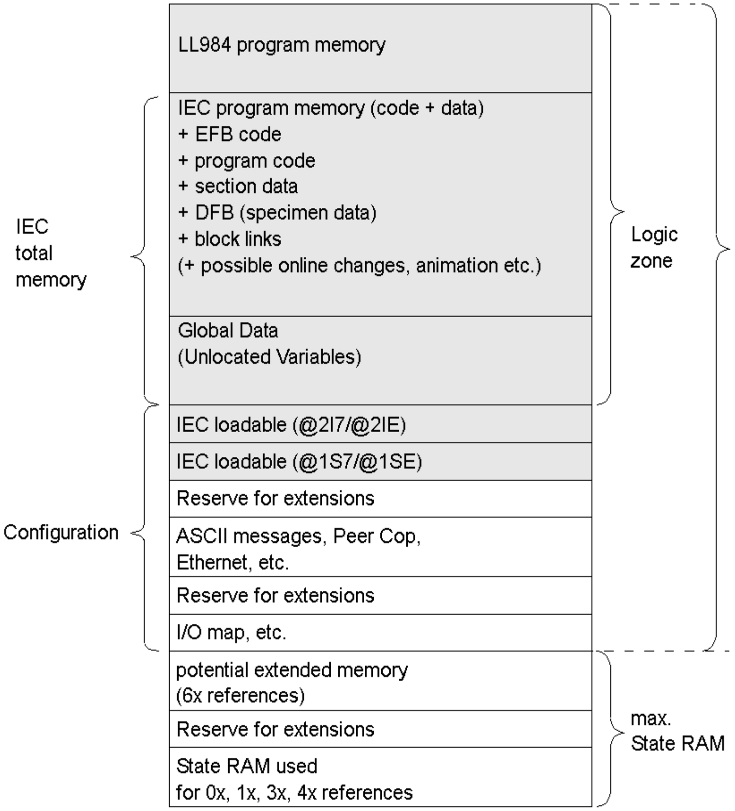
Introduction

If a CPU 213 04 or CPU 424 02 is used, you can make a zone in the state RAM available for the 6x references.

NOTE: 6x references are registers and can only be used with LL984 user programs.

Even if the state RAM memory zone is outside the program memory zone, the size of the state RAM influences the size of the program memory.

Using the extended memory (state RAM for 6x references):



If you do NOT use 6x

If you do not want to use any 6x references, you can, with a CPU 213 04, select whether to reserve state RAM 6x references or not.

Under **Project** →**PLC Configuration** →**PLC Selection** select from the **Memory Partition** the **48 K Logic / 32 K Memory**entry.

NOTE: With a CPU 424 02 there is no option for deactivating the 6x zone.

If you use 6x

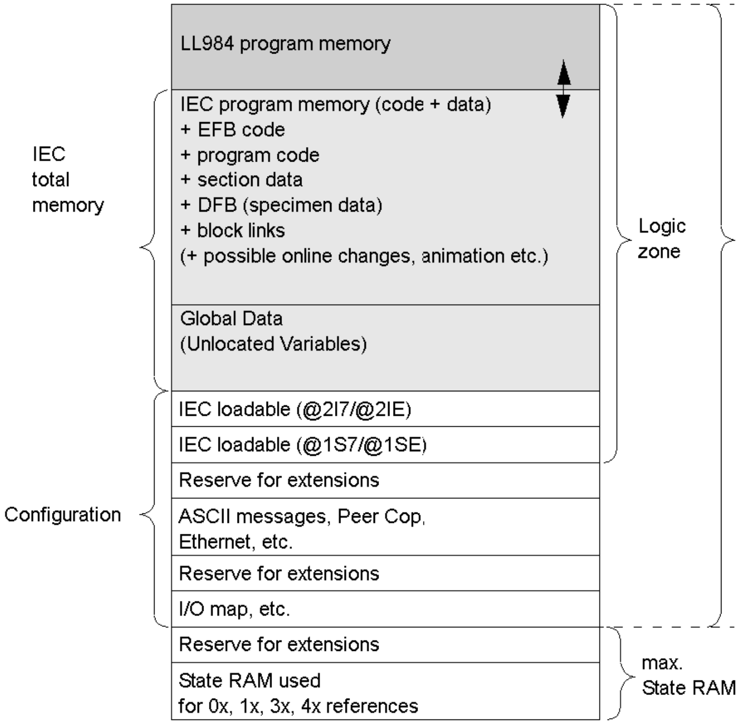
If you want to use 6x references, select under **Project** →**PLC Configuration** →**PLC selection** in the **Memory Partition** list box, the **32 K Logic / 64 K Memory**entry.

Harmonizing the IEC Zone and LL984 Zone

Introduction

With joint use of IEC and LL984 sections, the sizes of both zones should be harmonized with each other.

Harmonizing the IEC zone and LL984 zone:



Size of IEC Zone

The size of the total IEC memory and also the available space for LL984 data (user program) is determined by the memory size of the loadable @2I7 or @2IE.

You can define the memory size of the loadables in **Project →PLC Configuration →Loadables →Install @2I7 or @2IE →Edit... →Memory Size**.

The total size is given in paragraphs. A paragraph equals 16 bytes.

For the @1S7 or @1SE loadables, no memory size is needed. Ensure that "0" is specified here.

The fixed total IEC memory size is again made up of several zones. You will find the explanation of how to harmonize these zones vertically in the chapter *Harmonizing the Zones for Global Data and IEC Program Memory, page 173*.

Size of LL984 Zone

The size of the available memory for LL984 user programs is calculated using the following formula:

LL984 zone = available LL984 nodes – memory size of loadable @2I7/@2IE – size of loadables @2I7 or @2IE – size of loadables @1S7 or @1SE

When doing this calculation, it must be ensured that the size of the LL984 zone is node-oriented and the remaining instructions are byte-oriented.

Error Message during Download of Program

There are three possible causes for an error message, which says that the user program is too large for the PLC memory, appearing during download:

1. The memory is currently too small.
2. The loadable memory size is too small (see current chapter).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see chapter *Harmonizing the Zones for Global Data and IEC Program Memory, page 173*).

Harmonizing the Zones for Global Data and IEC Program Memory

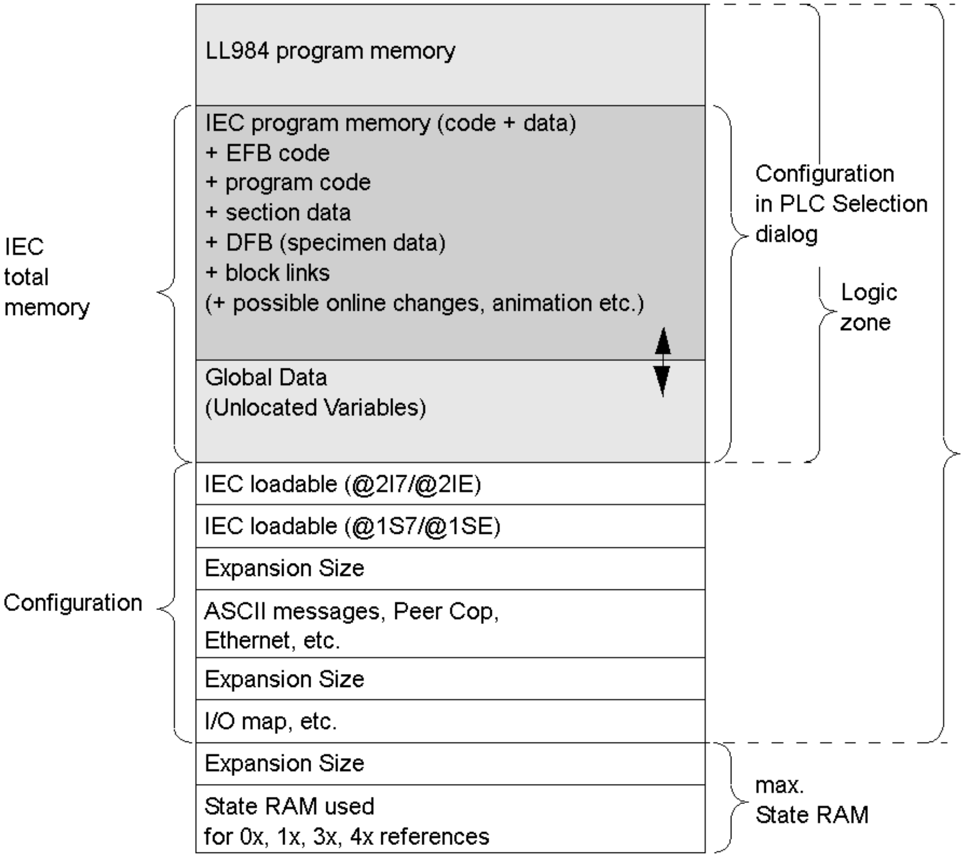
Introduction

The total IEC memory space, determined by the loadable memory size, (see Chapter *Harmonizing the IEC Zone and LL984 Zone, page 171*) is made up of two zones:

- **IEC Program Memory**
  - comprising the EFB codes,
  - the program codes,
  - the section data,
  - the DFB specimen data,
  - the block links,
  - possibly data from online changes,
  - possibly animation data etc.
- **Global Data**
  - comprising the Unlocated Variables

The zones for global data and IEC program memory can be harmonized with one another.

Harmonizing the Zones for IEC Program Memory and Global Data:



Size of the IEC Program Memory Zone

You change the settings for the IEC program memory in **Project →PLC Configuration →PLC selection** in the **IEC**zone. Enter the size of the total IEC memory and the global data, so that the IEC program memory size will be calculated (IEC program memory size = total IEC memory - global data). This setting is only possible when the PC and PLC are offline. If you do not use any or only a few unlocated variables and have no or only a few block links, you can select the IEC program memory as very large, because hardly any memory is needed for global data.

Size of the Zone for Global Data

The zone for global data (unlocated variables) is calculated using the following formula:

Zone for global data = memory size of the loadable - IEC program memory

The current content of the individual zones (EFBs, specimen data, user program etc.) is displayed under **Online** →**Memory statistics...** →**Memory statistics**. This display is only possible when the PC and PLC are online.

Error Message during Download of Program

There are three possible reasons for an error message, which says that the user program is too large for the PLC memory, appearing while downloading the program onto the PLC:

1. The memory is currently too small.
2. The loadable memory size is too small (see Chapter *Harmonizing the IEC Zone and LL984 Zone, page 171*).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see current chapter).

6.4 Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B)

Overview

This section describes the memory optimization for the Quantum CPUs 434 12(A) and 534 14(A/B).

What's in this Section?

This section contains the following topics:

Topic	Page
General Information on Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B)	177
Harmonizing IEC Zone and LL984 Zone	179
Harmonizing the Zones for Global Data and IEC Program Memory (CPU 434 12(A) / 534 14 (A/B))	184

General Information on Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B)

Logic Memory

The program memory zone, in which the user program is located, is called the logic zone. This zone therefore determines the maximum size of your user program.

The current size of the logic zone is displayed under **Project →PLC Configurator** in the configurations overview in the **PLC** zone. The memory size is given in nodes for LL984 (1 node equals 11 bytes) and in kilobytes for IEC.

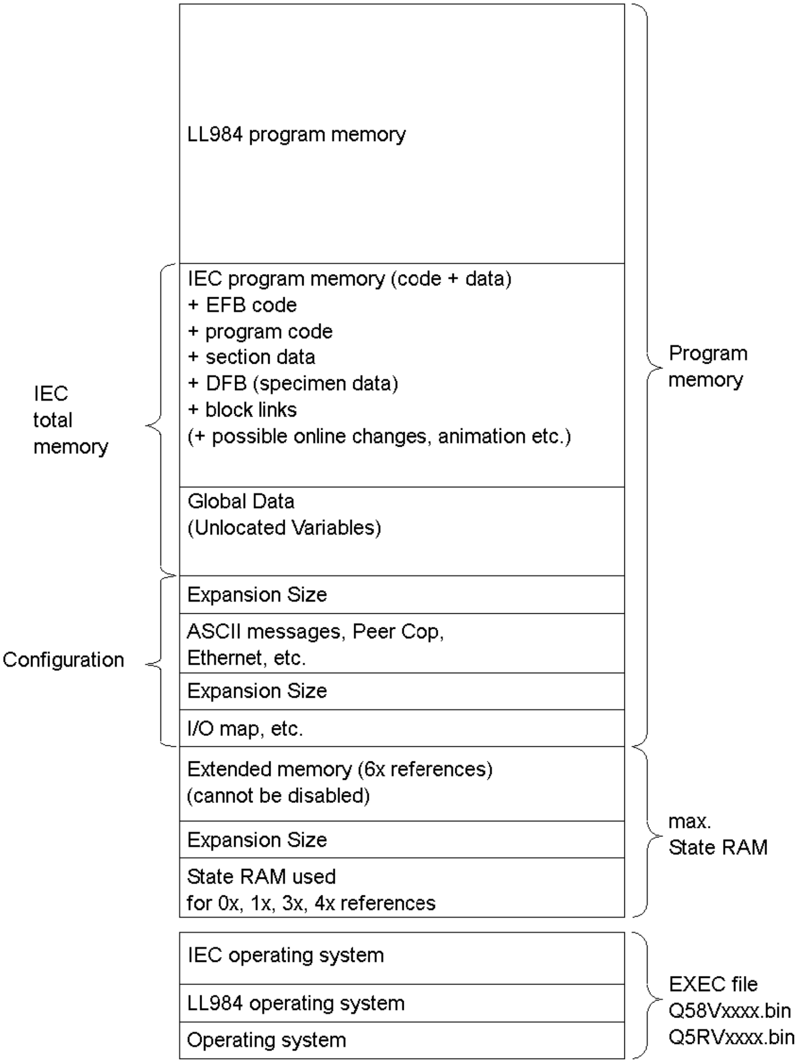
Optimizing the Logic Memory

You have various possibilities for optimising the logic memory to suit your requirements:

- *Harmonizing IEC Zone and LL984 Zone, page 179*
- *Harmonizing the Zones for Global Data and IEC Program Memory (CPU 434 12(A) / 534 14 (A/B)), page 184*

NOTE: Also note the PLC-independent possibilities for memory optimization *(see page 157)*.

Structure of the CPU 434 12(A) / 534 14(A/B) memory (simplified representation):



Harmonizing IEC Zone and LL984 Zone

Introduction

The EXEC file "Q58Vxxxx.bin" is required for the CPU 434 12 and 534 14.

The EXEC file "Q5RVxxxx.bin" is required for the CPU 434 12A and 534 14A/B (redesigned CPUs).

These EXEC files contain the runtime systems for IEC and LL984.

The sizes of the logic zones for IEC and LL984 should be harmonized with each other. The size of both zones can be defined in **Project** →**PLC Configurator** → **PLC selection**.

Depending on the size you select for the IEC zone, zones will be reserved in the program memory of the PLC for IEC and/or LL984 programs. Therefore, if you define a combined IEC and LL984 zone and then only use one of the two language types in the user program, the program memory will not be used optimally.

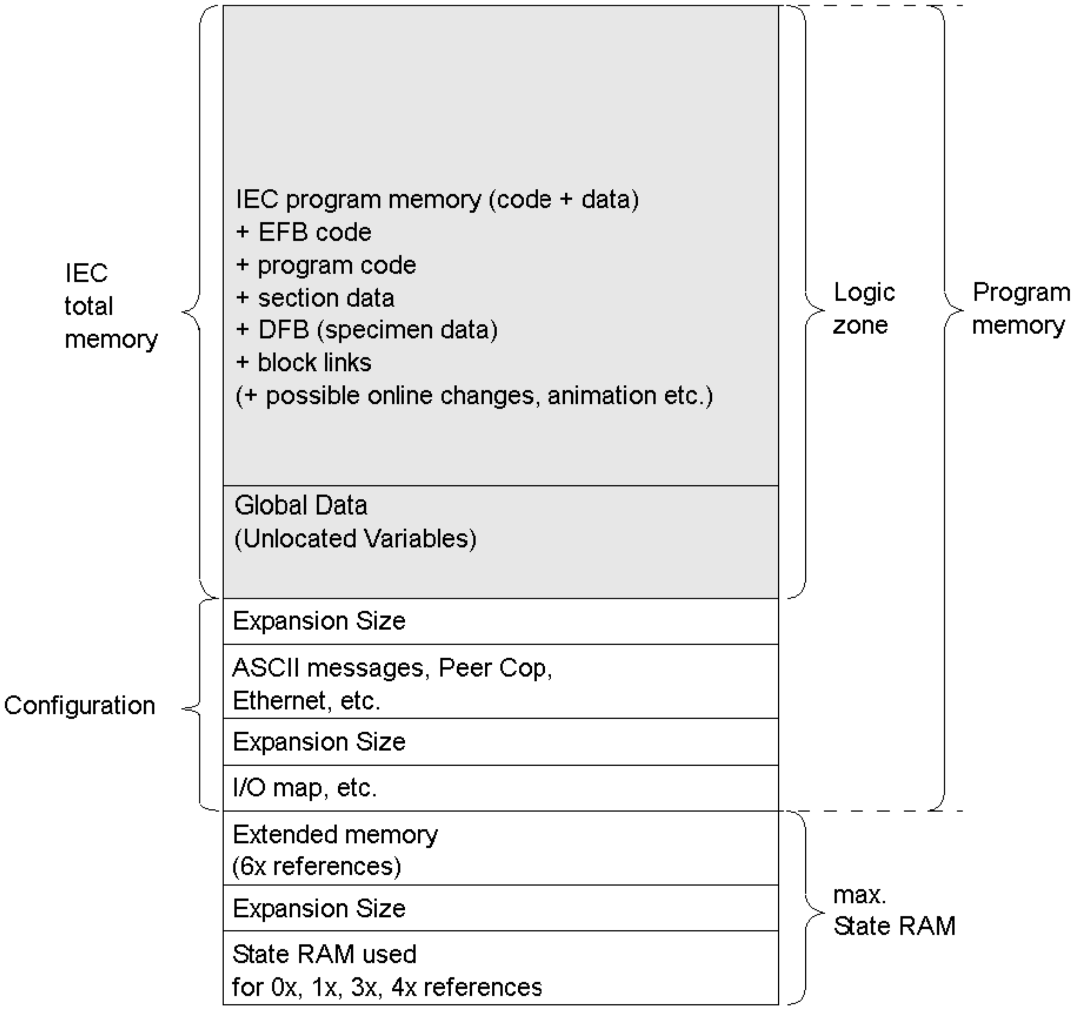
Therefore, decide which languages you want to use:

- *Exclusive Use of IEC, page 179*
- *Exclusive Use of LL984, page 180*
- *Joint Use of IEC and LL984, page 181*

Exclusive Use of IEC

If you require exclusive use of the IEC, select in **Project** →**PLC Configuration** → **PLC Selection** in the **IEC Operating System** list box, the entry **Enable** and drag the **total IEC memory** slider to the right hand margin (highest value). This will completely switch off the LL984 zone and the entire logic zone will be made available for the IEC user program.

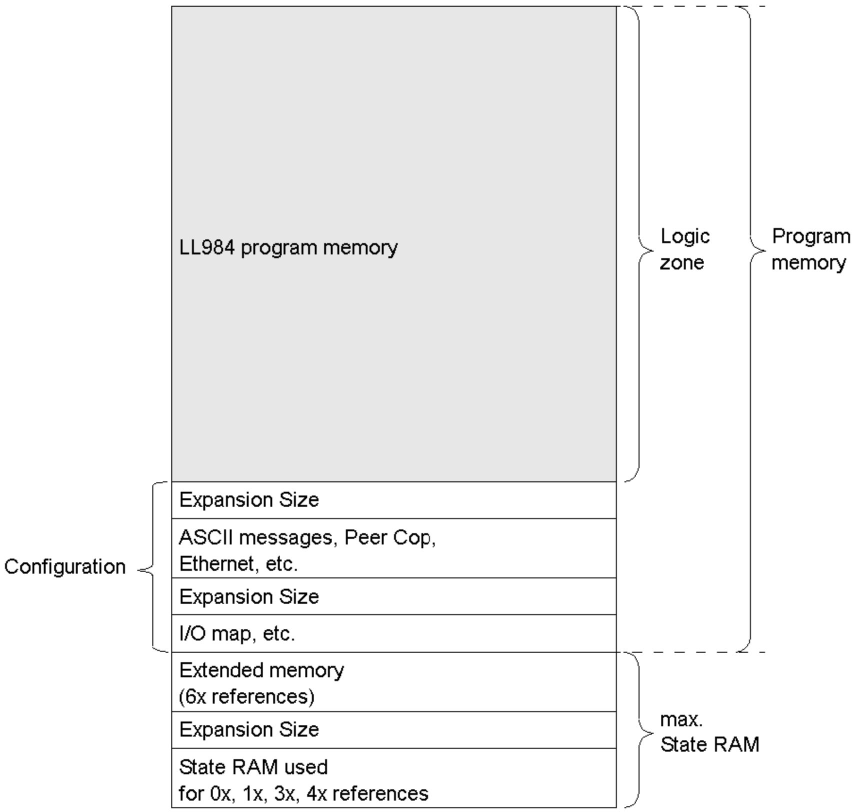
Structure of the CPU 434 12 (A)/ 534 14(A/B) memory with exclusive use of IEC:



Exclusive Use of LL984

If you require exclusive use of LL984, select from **Project** →**PLC Configuration** → **PLC Selection** in the **IEC Operating System** list box, the **Disable** entry. This will completely switch off the IEC zone and the entire logic zone will be made available for the LL984 user program.

Structure of the CPU 434 12(A)/ 534 14(A/B) memory with exclusive use of LL984:



Joint Use of IEC and LL984

When using IEC and LL984 jointly, you should harmonize the sizes of both zones with each other.

By setting the **total IEC memory size** and **Global Data** you can automatically determine the size of the IEC program memory, and also the available space for LL984-data (user program).

The size of the available memory for LL984 user programs is calculated using the following formula:

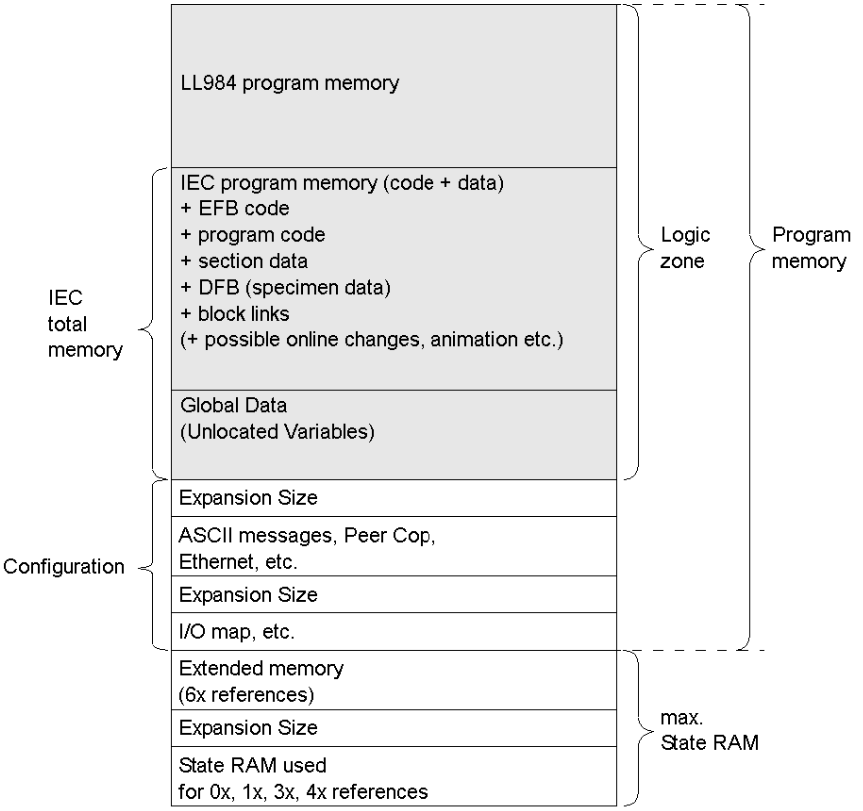
LL984 zone = available LL984 nodes - total IEC memory

When performing this calculation, it must however be ensured that the size of the LL984 zone is node-oriented and the remaining instructions are kilobyte-oriented.

To set the total IEC memory, select from **Project** →**PLC Configuration** →**PLC selection** in the **IEC Operating System** list box, the **Enable** entry. The IEC zone is now enabled and you can enter the required memory size in the **Total IEC Memory** text box. The memory size is given in kilobytes.

The fixed total IEC memory size is again made up of several zones. You will find the explanation of how to harmonize these zones vertically in the chapter *Harmonizing the Zones for Global Data and IEC Program Memory, page 173*.

Structure of the CPU 434 12(A)/ 534 14(A/B) memory with exclusive use of IEC and LL984:



Error Message during Download of Program

There are three possible causes for an error message, which says that the user program is too large for the PLC memory, appearing during download:

1. The memory is currently too small.
2. The logic zone is too small (see current chapter).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see chapter *Harmonizing the Zones for Global Data and IEC Program Memory (CPU 434 12(A) / 534 14 (A/B)), page 184*).

Harmonizing the Zones for Global Data and IEC Program Memory (CPU 434 12(A) / 534 14 (A/B))

Introduction

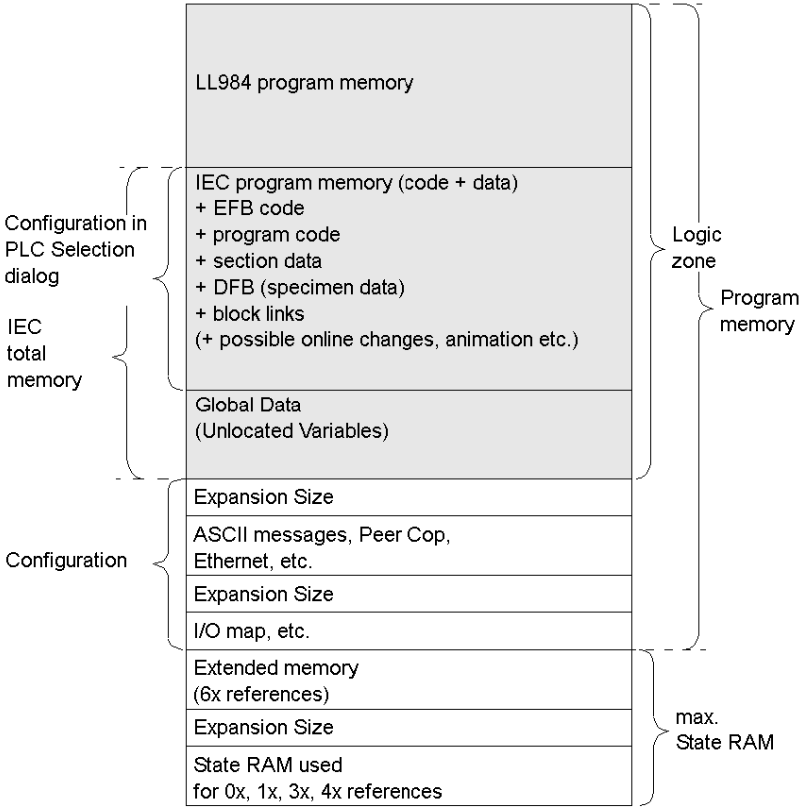
The fixed total IEC memory (see chapter *Harmonizing IEC Zone and LL984 Zone, page 179*) is made up of two zones.

The total IEC memory space, determined by the loadable memory size, (see Chapter *Harmonizing the IEC Zone and LL984 Zone, page 171*) is made up of two zones:

- **IEC Program Memory**
  - comprising the EFB codes,
  - the program codes,
  - the section data,
  - the DFB specimen data,
  - the block links,
  - possibly data from online changes,
  - possibly animation data etc.
- **Global Data**
  - comprising the Unlocated Variables

The zones for global data and IEC program memory can be harmonized with one another.

Harmonizing the Zones for Global Data and IEC Program Memory (CPU 434 12(A) / 534 14 (A/B))



Size of the IEC Program Memory Zone

You change the settings for the IEC program memory in **Project** →**PLC Configuration** →**PLC selection** in the **IEC**zone. Enter the size of the total IEC memory and the global data, so that the IEC program memory size will be calculated (IEC program memory size = total IEC memory - global data). This setting is only possible when the PC and PLC are offline. If you do not use any or only a few unlocated variables and have no or only a few block links, you can select the IEC program memory as very large, because hardly any memory is needed for global data.

Size of the Zone for Global Data

The zone for global data (unlocated variables) is calculated using the following formula:

Zone for global data = memory size of the loadable - IEC program memory

The current content of the individual zones (EFBs, specimen data, user program etc.) is displayed under **Online** →**Memory statistics...** →**Memory statistics**. This display is only possible when the PC and PLC are online.

Error Message during Download of Program

There are three possible reasons for an error message, which says that the user program is too large for the PLC memory, appearing while downloading the program onto the PLC:

1. The memory is currently too small.
2. The total IEC memory size is too small (see Chapter *Harmonizing IEC Zone and LL984 Zone, page 179*).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see current chapter).

6.5 Memory optimization for Compact CPUs

Overview

This Section describes the memory optimization for Compact CPUs.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information on Memory Optimization for Compact CPUs	188
Harmonizing IEC Zone and LL984 Zone	190
Harmonizing the Zones for Global Data and IEC Program Memory (Compact)	194

General Information on Memory Optimization for Compact CPUs

Logic Memory

The program memory zone, in which the user program is located, is called the logic zone. This zone therefore determines the maximum size of your user program.

The current size of the logic zone is displayed under **Project →PLC Configuration** in the configurations overview in the **PLC** zone. The entry for the memory size is given in Nodes for LL984 (1 node equals 11 bytes) and in kilobytes for IEC.

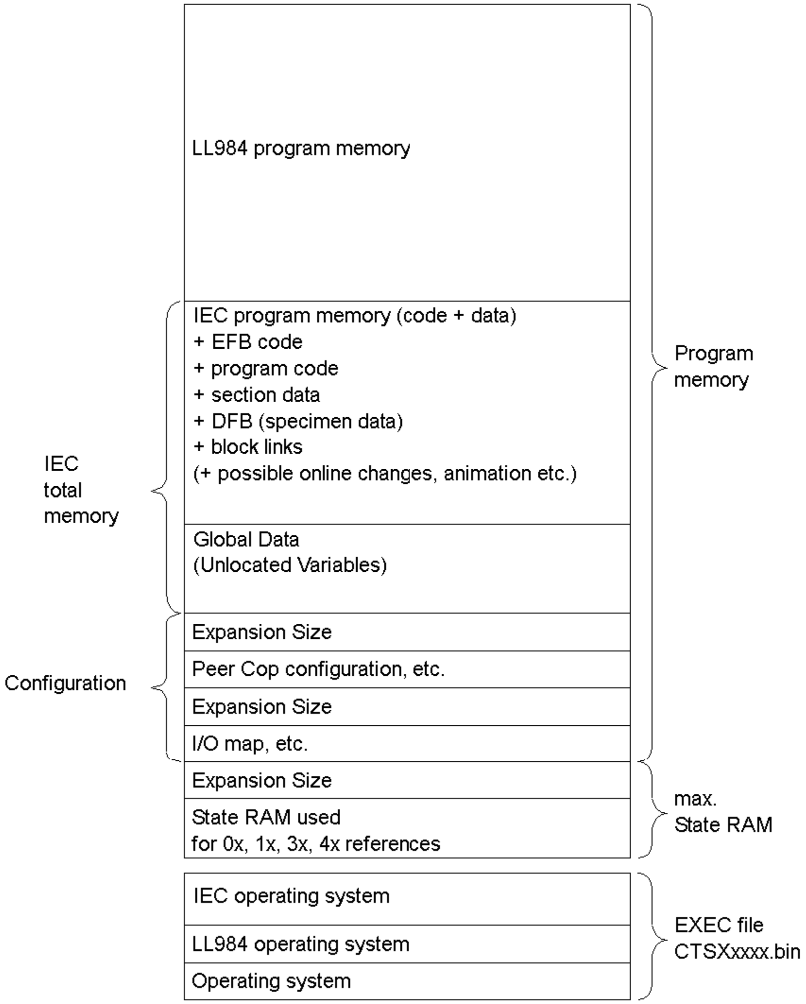
Optimizing the Logic Memory

You have various possibilities for optimising the logic memory to suit your requirements:

- *Harmonizing IEC Zone and LL984 Zone, page 190*
- *Harmonizing the Zones for Global Data and IEC Program Memory (Compact), page 194*

NOTE: Also note the PLC-independent possibilities for memory optimization *(see page 157)*.

Structure of a Compact CPU memory (simplified representation)



Harmonizing IEC Zone and LL984 Zone

Introduction

The IEC zone "CTSXxxxx.bin", required for Compact CPUs, contains the runtime systems for IEC and LL984 (see also *Installation instructions*).

The sizes of the logic zones for IEC and LL984 should be harmonized with each other. You can define the size of both zones in **Project** →**PLC Configurator** → **PLC Selection**.

Depending on the size you select for the IEC zone, zones will be reserved in the program memory of the PLC for IEC and/or LL984 programs. Therefore, if you define a combined IEC and LL984 zone and then only use one of the two language types in the user program, the program memory will not be used optimally.

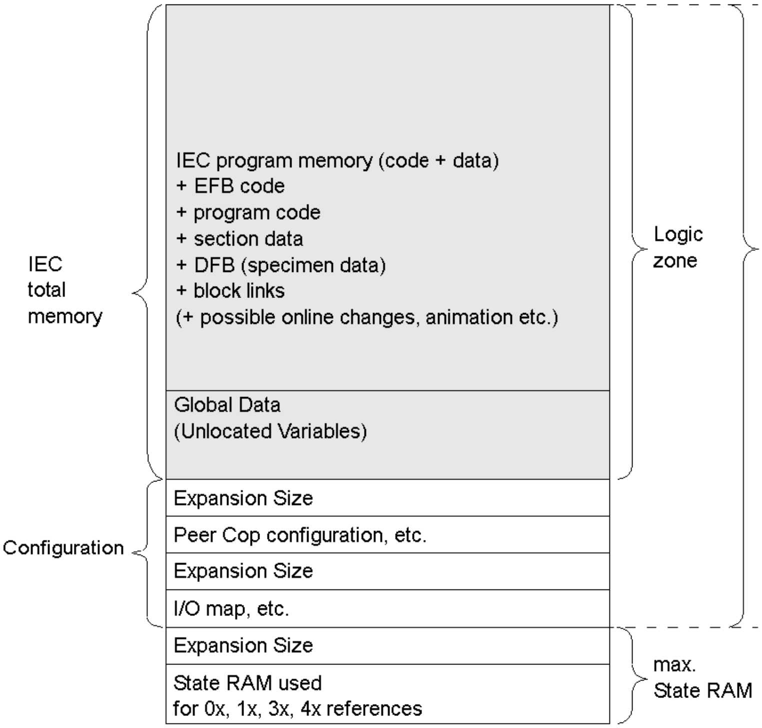
Therefore, decide which languages you want to use:

- *Exclusive Use of IEC, page 190*
- *Exclusive Use of LL984, page 191*
- *Joint Use of IEC and LL984, page 192*

Exclusive Use of IEC

If you require exclusive use of the IEC, select in **Project** →**PLC Configuration** → **PLC Selection** in the **IEC Operating System** list box, the entry **Enable** and drag the **total IEC memory** slider to the right hand margin (highest value). This will completely switch off the LL984 zone and the entire logic zone will be made available for the IEC user program.

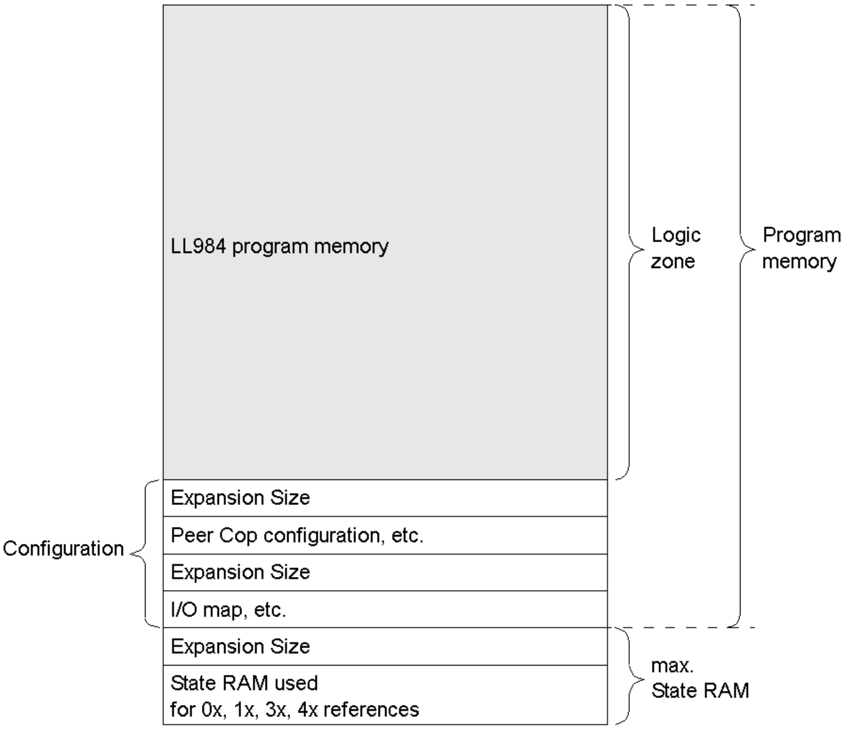
Structure of the Compact CPU memory with exclusive use of IEC



Exclusive Use of LL984

If you require exclusive use of LL984, select from **Project** →**PLC Configuration** → **PLC Selection** in the **IEC Operating System** list box, the **Disable** entry. This will completely switch off the IEC zone and the entire logic zone will be made available for the LL984 user program.

Structure of the Compact CPU memory with exclusive use of LL984



Joint Use of IEC and LL984

When using IEC and LL984 jointly, you should harmonize the sizes of both zones with each other.

By setting the **total IEC memory size** and **Global Data** you can automatically determine the size of the IEC program memory, and also the available space for LL984-data (user program).

The size of the available memory for LL984 user programs is calculated using the following formula:

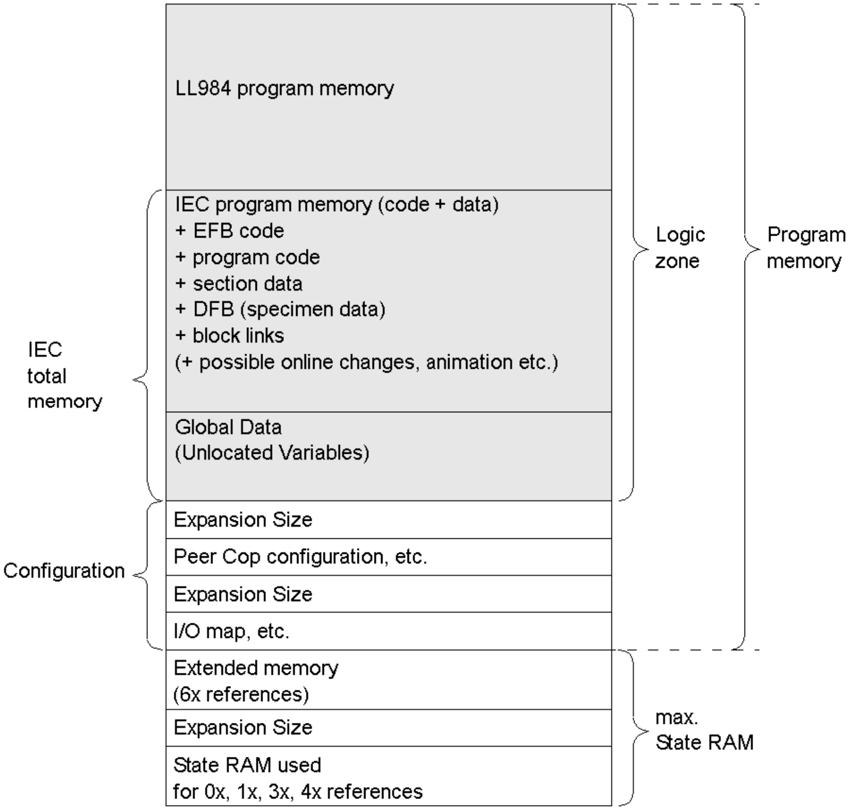
LL984 zone = available LL984 nodes - total IEC memory

When performing this calculation, it must however be ensured that the size of the LL984 zone is node-oriented and the remaining instructions are kilobyte-oriented.

To set the total IEC memory, select from **Project →PLC Configuration →PLC selection** in the **IEC Operating System** list box, the **Enable** entry. The IEC zone is now enabled and you can enter the required memory size in the **Total IEC Memory** text box. The memory size is given in kilobytes.

The fixed total IEC memory size is again made up of several zones. You will find the explanation of how to harmonize these zones vertically in the chapter *Harmonizing the Zones for Global Data and IEC Program Memory (Compact), page 194*.

Structure of the Compact Memory with joint use of IEC and LL984:



Error Message during Download of Program

There are three possible causes for an error message, which says that the user program is too large for the PLC memory, appearing during download:

1. The memory is currently too small.
2. The logic zone is too small (see current chapter).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see chapter *Harmonizing the Zones for Global Data and IEC Program Memory (Compact), page 194*).

Harmonizing the Zones for Global Data and IEC Program Memory (Compact)

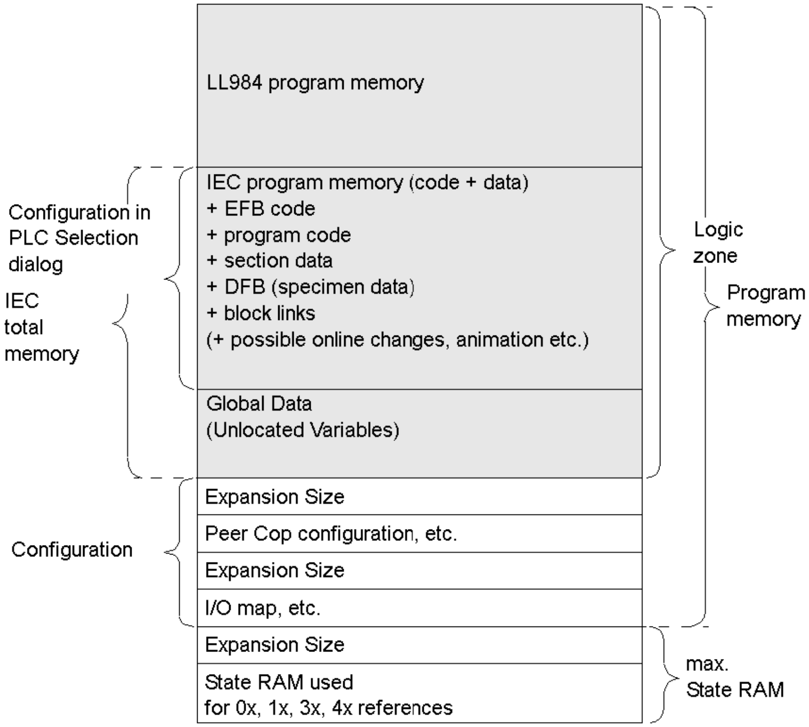
Introduction

The fixed total IEC memory (see chapter *Harmonizing IEC Zone and LL984 Zone, page 190*) is made up of two zones.

- **IEC Program Memory**
  - comprising the EFB codes,
  - the program codes,
  - the section data,
  - the DFB specimen data,
  - the block links,
  - possibly data from online changes,
  - possibly animation data etc.
- **Global Data**
  - comprising the Unlocated Variables

The zones for global data and IEC program memory can be harmonized with one another.

Harmonizing the Zones for Global Data and IEC Program Memory (Compact):



Size of the IEC Program Memory Zone

You change the settings for the IEC program memory in **Project** →**PLC Configuration** →**PLC selection** in the **IEC** zone. Enter the size of the total IEC memory and the global data, so that the IEC program memory size will be calculated (IEC program memory size = total IEC memory - global data). This setting is only possible when the PC and PLC are offline. If you do not use any or only a few unlocated variables and have no or only a few block links, you can select the IEC program memory as very large, because hardly any memory is needed for global data.

Size of the Zone for Global Data

The zone for global data (unlocated variables) is calculated using the following formula:

Zone for global data = memory size of the loadable - IEC program memory

The current content of the individual zones (EFBs, specimen data, user program etc.) is displayed under **Online** →**Memory statistics...** →**Memory statistics**. This display is only possible when the PC and PLC are online.

Error Message during Download of Program

There are three possible reasons for an error message, which says that the user program is too large for the PLC memory, appearing while downloading the program onto the PLC:

1. The memory is currently too small.
2. The total IEC memory size is too small (see Chapter *Harmonizing IEC Zone and LL984 Zone, page 190*).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see current chapter).

6.6 Memory optimization for Momentum CPUs

Overview

This Section describes the memory optimization for Momentum CPUs.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information on Memory Optimization for Momentum CPUs	198
Selecting Optimal EXEC file	200
Harmonizing the Zones for Global Data and IEC Program Memory (Momentum)	201

General Information on Memory Optimization for Momentum CPUs

Logic Memory

The program memory zone, in which the user program is located, is called the logic zone. This zone therefore determines the maximum size of your user program.

The current size of the logic zone is displayed under **Project →PLC Configuration** in the configurations overview in the **PLC** zone. The entry for the memory size is given in Nodes for LL984 (1 node equals 11 bytes) and in kilobytes for IEC.

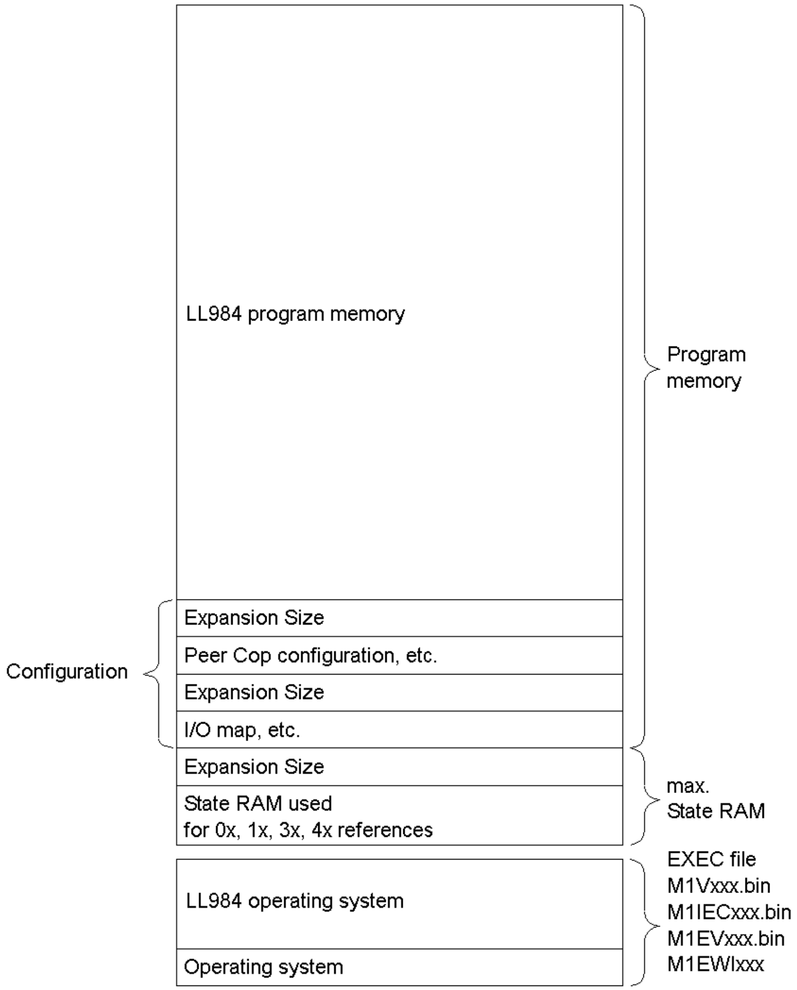
Optimizing the Logic Memory

You have various possibilities for optimising the logic memory to suit your requirements:

- *Selecting Optimal EXEC file, page 200*
- *Harmonizing the Zones for Global Data and IEC Program Memory (Momentum), page 201*

NOTE: Also note the PLC-independent possibilities for memory optimization *(see page 157)*.

Structure of a Momentum CPU memory (simplified representation):



Selecting Optimal EXEC file

Introduction

It is not possible to use IEC and LL984 jointly in Momentum.

Using IEC

EXEC file assignment during IEC use:

171 CBB	M1IVxxxE	MPSV100e.BIN
970 30	-	x

171 CCS	M1IVxxxE	M1EVxxxE
760 00	x	-
760 10	x	-
780 10	x	-
960 30	-	x
980 30	-	x

Using LL984

EXEC file assignment during LL984 use:

171 CBB	M1LLVxxx	M1MVxxxE
970 30	x	-

171 CCS	M1LLVxxx	M1EVxxx
700 10	x	-
700/780 00	x	-
760 00	x	-
760 10	x	-
780 10	x	-
960 20	-	x
960 30	-	x
980 20	-	x
980 30	-	x

Harmonizing the Zones for Global Data and IEC Program Memory (Momentum)

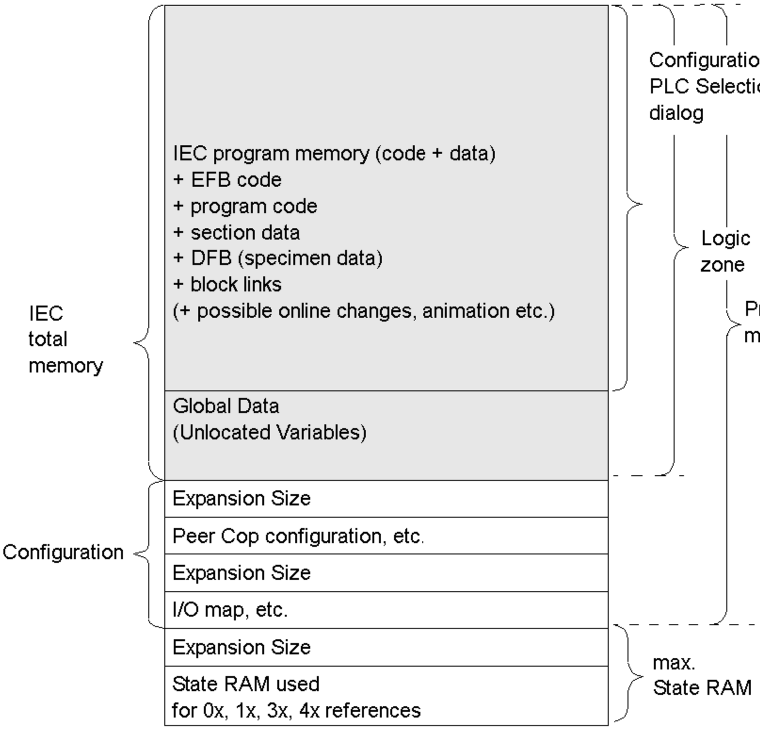
Introduction

The logic zone for the total IEC memory is made up of two zones.

- **IEC Program Memory**
  - comprising the EFB codes,
  - the program codes,
  - the section data,
  - the DFB specimen data,
  - the block links,
  - possibly data from online changes,
  - possibly animation data etc.
- **Global Data**
  - comprising the Unlocated Variables

The zones for global data and IEC program memory can be harmonized with one another.

Harmonizing the Zones for Global data and IEC Program Memory (Momentum 171 CCS 760 00-IEC):



Size of the IEC Program Memory Zone

The settings for the IEC user program zone are available in **Online** →**Memory statistics...** →**Memory statistics** in the **Configured** text box. This setting is only possible when the PC and PLC are offline. If you do not use any or only a few unlocated variables and have no or only a few block links, you can select the IEC program memory as very large, because hardly any memory is needed for global data.

Size of the Zone for Global Data

The zone for global data (unlocated variables and block links) is calculated using the following formula:

Zone for global data = memory size of the loadable - IEC program memory

The current content of the individual zones (EFBs, specimen data, user program etc.) is displayed under **Online** →**Memory statistics...** →**Memory statistics**. This display is only possible when the PC and PLC are online.

Error Message during Download of Program

There are two possible reasons for an error message, saying that the user program is too large for the PLC memory, appearing while downloading the program onto the PLC:

1. The memory is currently too small.
2. The zone for global data and the IEC program memory zone are not optimally harmonized (see current chapter).

6.7 Memory optimization for Atrium CPUs

Overview

This Section describes the memory optimization for Atrium CPUs.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information on Memory Optimization for Atrium CPUs	205
Use of IEC	207
Harmonizing the Zones for Global Data and IEC Program Memory (Atrium)	209

General Information on Memory Optimization for Atrium CPUs

Logic Memory

The program memory zone, in which the user program is located, is called the logic zone. This zone therefore determines the maximum size of your user program.

The current size of the logic zone is displayed under **Project →PLC Configurator** in the configurations overview in the **PLC** zone. The memory size is given in kilobytes for IEC.

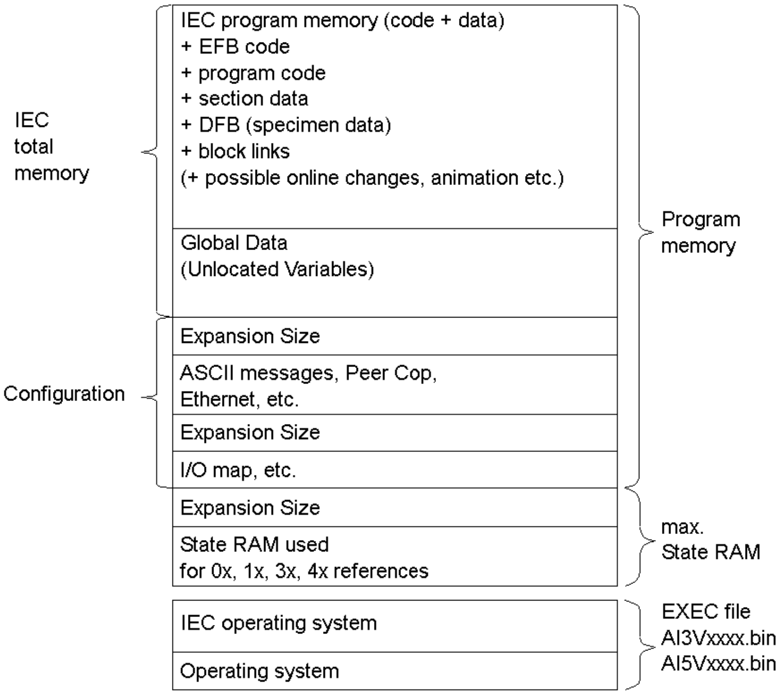
Optimizing the Logic Memory

You have various possibilities for optimising the logic memory to suit your requirements:

- *Use of IEC, page 207*
- *Harmonizing the Zones for Global Data and IEC Program Memory (Atrium), page 209*

NOTE: Also note the PLC-independent possibilities for memory optimization *(see page 157)*.

Structure of the Atrium CPU Memory (simplified representation):



Use of IEC

Introduction

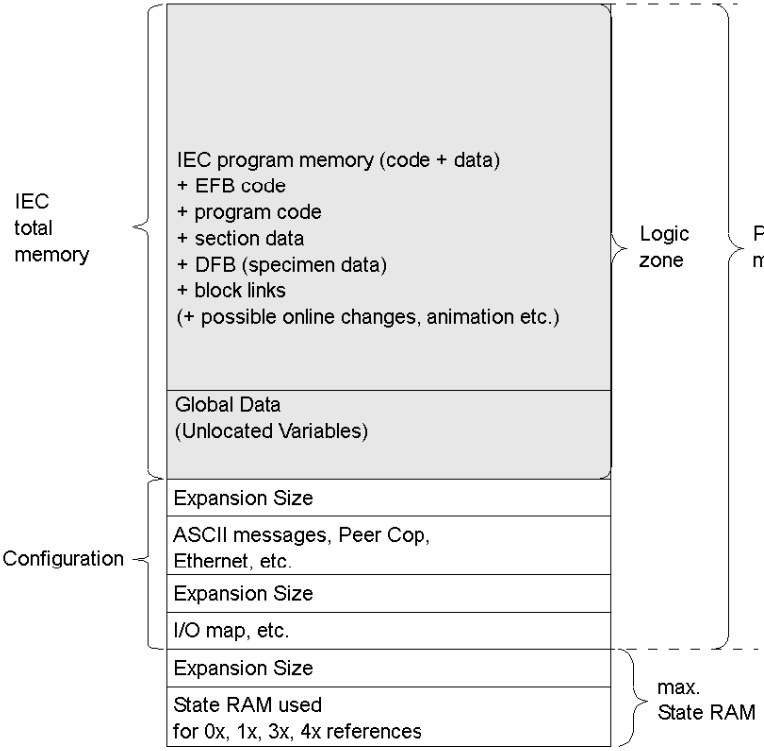
The EXEC files required for the CPUs of the Atrium family contain the operating systems for IEC (see also *Installation Instructions*).

When using the Atrium 180 CCO 121 01, load the EXEC file "AI3Vxxxx.bin".

When using the Atrium 180 CCO 241 01and 180 CCO 241 11 load the EXEC file "AI5Vxxxx.bin".

Select in **Project** →**PLC Configuration** →**PLC Selection** in the **IEC Operating System** list box, the entry **Enable** and drag the **total IEC memory** slider to the right hand margin (highest value). This will completely switch off the LL984 zone and the entire logic zone will be made available for the IEC user program.

Structure of the Atrium CPU memory with exclusive use of IEC:



Error Message during Download of Program

There are three possible causes for an error message, which says that the user program is too large for the PLC memory, appearing during download:

1. The memory is currently too small.
2. The logic zone is too small (see current chapter).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see chapter *Harmonizing the Zones for Global Data and IEC Program Memory (Atrium), page 209*).

Harmonizing the Zones for Global Data and IEC Program Memory (Atrium)

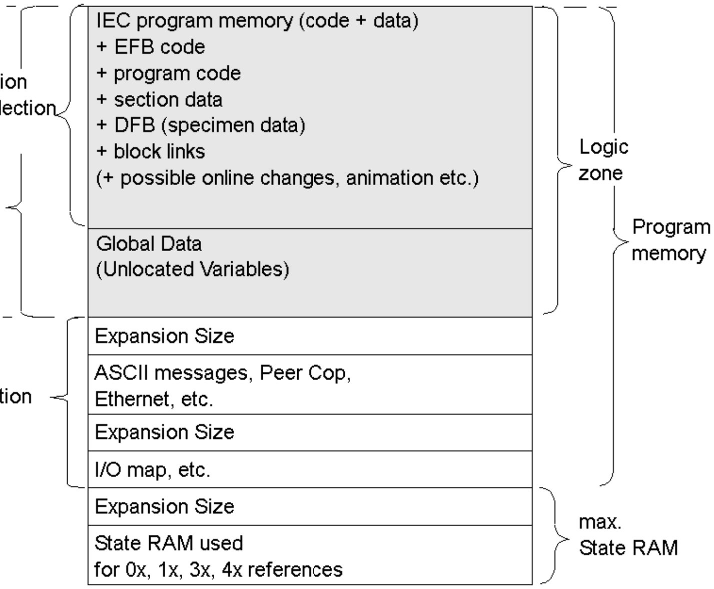
Introduction

The fixed total IEC memory (see chapter *Use of IEC, page 207*) is made up of two zones.

- **IEC Program Memory**
  - comprising the EFB codes,
  - the program codes,
  - the section data,
  - the DFB specimen data,
  - the block links,
  - possibly data from online changes,
  - possibly animation data etc.
- **Global Data**
  - comprising the Unlocated Variables

The zones for global data and IEC program memory can be harmonized with one another.

Harmonizing the Zones for Global Data and IEC Program Memory (Atrium):



Size of the IEC Program Memory Zone

You change the settings for the IEC program memory in **Project →PLC Configuration →PLC selection** in the **IEC** zone. Enter the size of the total IEC memory and the global data, so that the IEC program memory size will be calculated (IEC program memory size = total IEC memory - global data). This setting is only possible when the PC and PLC are offline. If you do not use any or only a few unlocated variables and have no or only a few block links, you can select the IEC program memory as very large, since hardly any memory is needed for global data.

Size of the Zone for Global Data

The zone for global data (unlocated variables) is calculated using the following formula:

Zone for global data = memory size of the loadable - IEC program memory

The current content of the individual zones (EFBs, specimen data, user program etc.) is displayed under **Online** →**Memory statistics...** →**Memory statistics**. This display is only possible when the PC and PLC are online.

Error Message during Download of Program

There are three possible reasons for an error message, which says that the user program is too large for the PLC memory, appearing while downloading the program onto the PLC:

1. The memory is currently too small.
2. The total IEC memory size is too small (see Chapter *Use of IEC, page 207*).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see current chapter).

Function Block language FBD

7

Overview

This Chapter describes the Function Block language FBD which conforms to IEC 1131.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
7.1	General information about FBD Function Block	214
7.2	FBD Function Block objects	215
7.3	Working with the FBD Function Block langauge	224
7.4	Code generation with the FBD Function Block language	232
7.5	Online functions of the FBD Function Block language	233
7.6	Creating a program with the FBD Function Block language	235

7.1 General information about FBD Function Block

General information on Function Block language FBD

At a Glance

The objects of the programming language FBD (Function Block Diagram) help to divide a section into a number of:

- EFBs (Elementary Functions and Elementary Function Blocks) *(see page 216)*,
- DFBs (Derived Function Blocks) *(see page 218)* and
- UDEFBs (User-defined Functions and Function Blocks) *(see page 219)*.

These objects, combined under the name FFBs, can be linked with each other by:

- Links *(see page 220)* or
- Current parameters *(see page 221)*.

Expansive logic can also be placed in the FBD section in the form of macros (see also *Macros, page 521*).

Theoretically, each section can contain as many FFBs and also as many inputs and outputs as required. However, it is advisable to subdivide a whole program in logic units, that is to say in different sections.

Comments can be provided for the logic of the section with text objects (see *Text Object, page 223*).

Processing sequence

The processing sequence of the individual FFBs in an FBD section is determined by the data flow within the section (see also *FFB Execution Order, page 227*).

Editing with the keyboard

Normally editing in Concept is performed with the mouse, however it is also possible with the keyboard (see also *Short Cut Keys in the FBD and SFC Editor, page 837*)

IEC conformity

For a description of the IEC conformity of the FBD programming language see *IEC conformity, page 849*.

7.2 FBD Function Block objects

Overview

This section describes the FBD Function Block objects.

What's in this Section?

This section contains the following topics:

Topic	Page
Functions and Function Blocks (FFBs)	216
Link	220
Actual parameters	221
Text Object	223

Functions and Function Blocks (FFBs)

Introduction

FFB is the generic term for:

- EFB (Elementary Function and Elementary Function Block) *(see page 216)*
- DFB (Derived Function Block) *(see page 218)*
- UDEFB (Derived Elementary Function and Derived Elementary Function Block) *(see page 219)*

EFB

EFB is the generic term for:

- Elementary Function *(see page 216)*
- Elementary Function Block *(see page 217)*

EFBs are functions and function blocks that are available in Concept in the form of libraries. The logic of EFBs is built in C programming language and cannot be changed in the FBD editor.

Elementary Function

Functions have no internal conditions. If the input values are the same, the value at the output is the same for all executions of the function. E.g. the addition of two values gives the same result at every execution.

An Elementary Function is represented graphically as a frame with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The name of the function, that is the function type, is displayed in the center of the frame. The function counter is displayed above the frame.

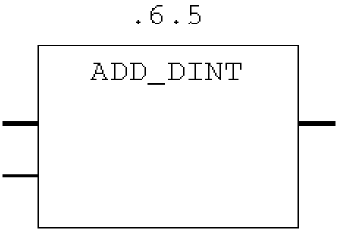
The function counter cannot be changed and always has an .n.m. structure.

.n = current section number

.m = current function number

Functions are only executed in FBD if the input EN=1 or if the input EN is grayed out (see also *EN and ENO, page 219*).

Elementary Function



Elementary Function Block

Function blocks have internal conditions. If the inputs have the same values, the value at the output at every execution is another value. E.g. with a counter, the value on the output is incremented.

A function block is represented graphically as a frame with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The name of the function block, that is the function block type, is displayed in the center of the frame. The instance name is displayed above the frame. The instance name serves as a unique identification for the function block in a project.

The instance name is produced automatically with the following structure: FBI_n_m

FBI = Function Block Instance

n = Section number (current number)

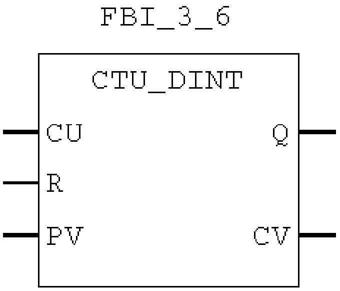
m = Number of the FFB object in the section (current number)

The instance name can be edited in the **Object** →**Properties** dialog box of the function block. The instance name must be unique throughout the whole project and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must correspond to the IEC name conventions, otherwise an error message occurs.

NOTE: In compliance with IEC1131-3 only letters are permitted as the first character of instance names. Should numbers be required as the first character however, the menu command **Options** →**Preferences** →**IEC Extensions...** →**Permit Leading Figures in Identifiers** will enable this.

Function blocks are only executed in FBD if the input EN=1 or if the input EN is grayed out (related topics *EN and ENO, page 219*).

Elementary Function Block



DFB

Derived Function Blocks (DFBs) are function blocks that have been defined in Concept DFB.

With DFBs, there is no distinction between functions and function blocks. They are always treated as function blocks regardless of their internal structure.

A DFB is represented graphically as a frame with double vertical lines and with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The DFB name is displayed centrally within the frame. The instance name is displayed above the frame. The instance name serves as a unique identification for the function block in a project.

The instance name is produced automatically with the following structure: FBI_n_m

FBI = Function Block Instance

n = Section number (current number)

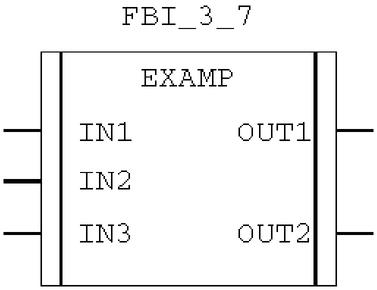
m = Number of the FFB object in the section (current number)

The instance name can be edited in the **Object** →**Properties** dialog box of the DFB. The instance name must be unique throughout the whole project and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must correspond to the IEC name conventions, otherwise an error message occurs.

NOTE: In compliance with IEC1131-3 only letters are permitted as the first character of instance names. Should numbers be required as the first character however, the menu command **Options** →**Preferences** →**IEC Extensions...** →**Permit Leading Figures in Identifiers** will enable this.

Derived function blocks are only executed in FBD if the input EN=1 or if the input EN is grayed out (related topics *EN and ENO, page 219*).

Derived Function Block



UDEFB

UDEFB is the generic term for:

- User-defined Elementary Function
- User-defined Elementary Function Block

UDEFBs are functions and function blocks that have been programmed with Concept EFB in C++ programming language and are available in Concept in the form of libraries.

In Concept, there is no functional difference between UDEFBs and EFBs.

EN and ENO

With all FFBs, an EN input and an ENO output can be configured.

The configuration of EN and ENO is switched on or off in the **FFB Properties** dialog box. The dialog box can be called up with the **Objects →Properties...** menu command or by double-clicking on the FFB.

If the value of EN is equal to "0" when the FFB is invoked, the algorithms that are defined by the FFB will not be executed and all outputs keep their previous values. The value of ENO is automatically set to "0" in this case.

If the value of EN is equal to "1", when the FFB is called up, the algorithms which are defined by the FFD will be executed. After successful execution of these algorithms, the value of ENO is automatically set to "1". If an error occurs during execution of these algorithms, ENO will be set to "0".

The output behavior of the FFBs in FBD does not depend on whether the FFBs are called up without EN/ENO or with EN=1.

Link

Description

Links are connections between FFBs.

Several links can be connected with one FFB output. The link points are identified by a filled-in circle.

Data Types

The data types of the inputs/outputs to be linked must be the same.

Creating Links

Links can be created using **Objects →Link**.

Editing Links

Links can be edited in select mode. An overlap with other objects is permitted.

Configuring Loops

No loop can be configured with links because in this case, the execution order in the section cannot be determined uniquely. Loops must be resolved with actual parameters (see *Configuring Loops, page 230*).

Actual parameters

At a Glance

In the program runtime, the values from the process or from other actual parameters are transferred to the FFB over the actual parameters and then re-emitted after processing.

These actual parameters can be:
- direct addresses *(see page 67)*
- Located variables *(see page 64)*
- Unlocated variable *(see page 64)*
- Constants *(see page 66)*
- Literals *(see page 66)*

Direct addresses

The information on/display of direct addresses can be given in various formats. The display format is set in the dialog box **Options** →**Presettings** →**Joint**. Setting the display format has no impact on the entry format, i.e. direct addresses can be entered in any format.

The following address formats are possible:
- **Standard format (400001)**
  The five-character address comes directly after the first digit (the Reference).
- **Separator format (4:00001)**
  The first digit (the Reference) is separated from the following five-character address by a colon (:).
- **Compact format (4:1)**
  The first digit (the Reference) is separated from the following address by a colon (:), and the leading zeros of the address are not given.
- **IEC format (QW1)**
  In first place, there is an IEC identifier, followed by the five-character address.
  - %0x12345 = %Q12345
  - %1x12345 = %I12345
  - %3x12345 = %IW12345
  - %4x12345 = %QW12345

Data types

The data type of the actual parameter must match the data type of the input/output. The only exceptions are generic inputs/outputs, of which the data type is determined by the formal parameter. If all actual parameters consist of literals, a suitable data type is selected for the Function Block.

Initial values

FFBs, which use actual parameters on the inputs that have not yet received any value assignment, work with the initial values of these actual parameters.

Unconnected inputs

NOTE: Unconnected FFB inputs are specified as "0" by default.

Text Object

At a Glance

Text can be positioned in the form of text objects using FBD Function Block language. The size of these text objects depends on the length of the text. The size of the object, depending on the size of the text, can be extended vertically and horizontally to fill further grid units. Text objects may not overlap with FFBs; however they can overlap with links.

Memory space

Text objects occupy no memory space on the PLC because the text is not downloaded onto the PLC.

7.3 Working with the FBD Function Block langauge

Overview

This section describes working with the FBD Function Block object language..

What's in this Section?

This section contains the following topics:

Topic	Page
Positioning Functions and Function Blocks	225
FFB Execution Order	227
Configuring Loops	230

Positioning Functions and Function Blocks

Selecting FFBs

Using **Objects** →**Select FFB...** you can open a dialog for selecting FFBs. This dialog is modeless, that is, it is not automatically closed once an FFB is positioned, but remains open until you close it. If you have several FBD sections open, and invoke the dialog, only one dialog box is opened that is available for all sections. The dialog box is not available for any other sections (non-FBD editor). If the FBD sections are changed into icons (minimize window), the dialog box is closed. If one of the FBD section icons is called up again, the dialog box is automatically re-opened.

The first time Concept is started the FFB is displayed oriented to the library. This means that, when selecting an FFB, the **Library** command button must first of all be used to select the corresponding library. Then you can select the corresponding **Group** in the list box. Now, you can select the required FFB from the EFB type list.

If you do not know which library/group the FFB required is in, you can invoke an FFB-oriented dialog with the **Sorted by FFB** command button. This contains all FFBs of all libraries and groups in an alphabetical list.

After each subsequent project start, the view you selected appears.

Once the FFB has been selected, its position in the section must be selected. The cursor becomes a small FFB and the cross shows the position (upper left corner of the FFB) where the FFB is positioned. The FFB is positioned by clicking on the left-hand mouse button.

Positioning FFBs (Functions and Function Blocks)

In the FBD function block language editor, the window appears with a logic grid. FFBs *(see page 216)* are aligned in this grid as they are positioned. If FFBs are positioned outside of the section frame or if there is overlapping with another FFB, an error warning will appear and the FFB will not be positioned. Actual parameters may overlap another object when being positioned at an FFB input/output, but they must not go outside the limits of the section frame. If a link to another FFB is established, this link is checked. If this link is not permitted, a message is received, and the link is not established. When links are created, overlaps and crossing with other links and FFBs are permitted. If an FFB is selected, the comment relating to it is displayed in the first column of the status line. If an actual parameter is selected, its name and, if applicable, its direct address, its I/O map and its comment are displayed in the first column of the status line.

Change FFB Type

With the **Objects →Replace FFBs...** menu command the FFBs already positioned in the section can be replaced with FFBs of another type (e.g. an AND with an OR). The variables given to the FFB remain if the data type and position of the inputs/outputs are the same as the "old" and the new FFB.

NOTE: FFBs with inputs / outputs of the ANY data type (generic FFBs) cannot be replaced.

FFB Execution Order

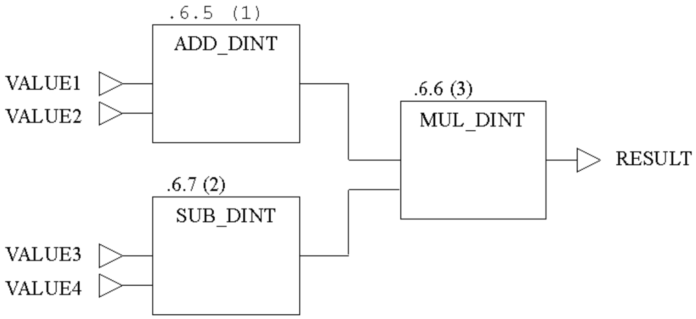
Introduction

The execution order is first determined by the order when positioning the FFB. If the FFBs are then linked graphically, the execution order is determined by the data flow.

Display FFB Execution Order.

The execution order can also be displayed with the **Objects →FFB Execution Order** menu command. This is represented by the execution number (number in brackets behind the instance name or function counter).

Show execution order of the FFBs



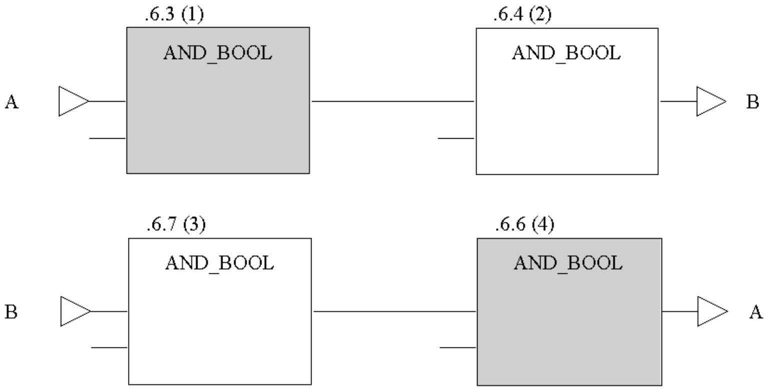
Change FFB Execution Order

The execution order can be specifically changed afterwards with the menu command **Objects →Change FFB Execution Order**, but only if the rules regarding data flow are not broken.

Changing the execution order of two networks which are in one loop

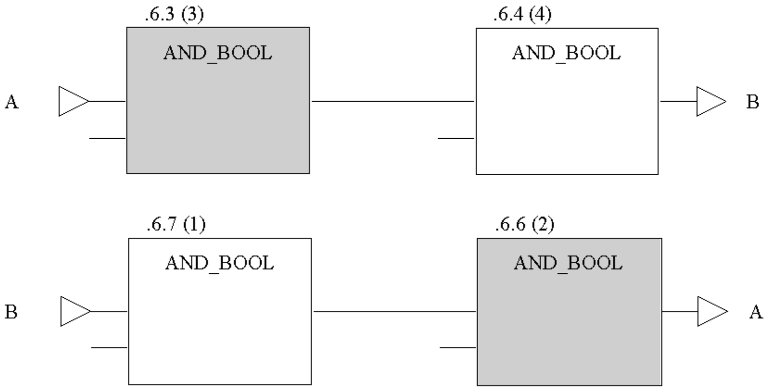
This change can only be made when the two FFBs are linked by the feedback variable of the loop.

Step 1: Select the two FFBs.



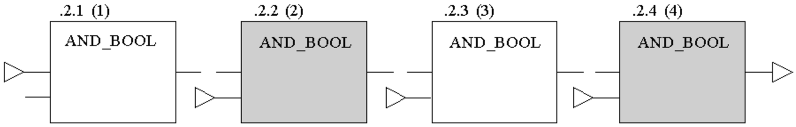
Step 2: Press the menu command **Change FFB-execution sequence**.

Result: The execution sequence has changed as follows:

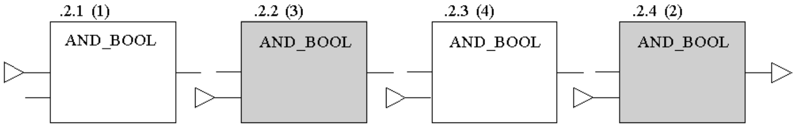


Changing the execution order of FFBs which are executed according to the positioning order

The change operation permits the creation of a different, desired order (sometimes step by step if several FFBs are involved).



Result: The execution sequence has changed as follows:

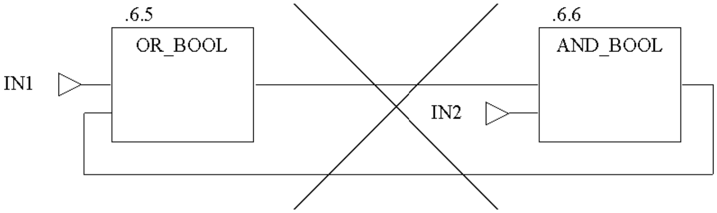


Configuring Loops

Non-permitted Loops

Configuring loops exclusively via links is not permitted, as it is not possible to uniquely set the data flow (the output of one FFB is the input of the next FFB, and the output of this one is the input of the first).

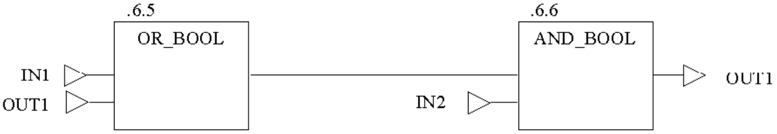
Non-permitted Loops via Links



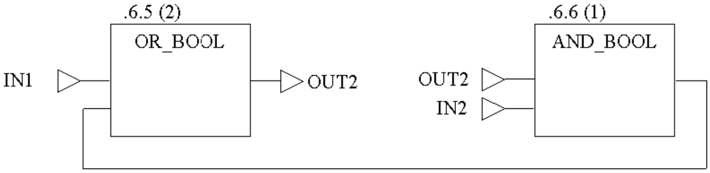
Resolution using an Actual Parameter

This type of logic must be resolved using actual parameters so that the data flow can be determined uniquely.

Resolved loop using an actual parameter: Variant 1



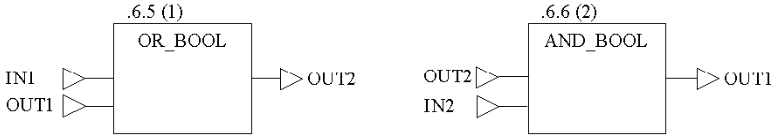
Resolved loop using an actual parameter: Variant 2



Resolution using Several Actual Parameters

Loops using several actual parameters are also allowed. With such loops, the execution order can later be influenced by executing – possibly several times – the menu command **Objects →Reverse FFB Execution Order** (see also *FFB Execution Order, page 227*).

Loop using several actual parameters



7.4 Code generation with the FBD Function Block language

Code Generation Options

Introduction

Using the **Project** →**Code Generation Options** menu command, you can define options for code generation.

Include Diagnosis Information

If the **Include Diagnosis information** check box is checked, additional information for the process diagnosis (e.g. Transition Diagnosis *(see page 319)*, diagnosis codes for diagnosis function blocks with extended diagnosis, such as e.g. XACT, XLOCK etc.) will be produced during code generation. This process diagnosis can be evaluated with MonitorPro or FactoryLink, for example.

Fastest Code (Restricted Checking)

If you check the **Fastest code (Restricted Checking)** check box, a runtime-optimized code is generated. This runtime optimization is achieved by realizing the integer arithmetic (e.g. "+" or "-") using simple CPU commands instead of EFB invocations.

CPU commands are much quicker than EFB invocations, but they do not generate any error messages, such as, for example, arithmetic or array overflow. This option should only be used when you have ensured that the program is free of arithmetic errors.

If **Fastest Code (Restricted Checking)** was selected, the addition IN1 + 1 is solved with the "add" CPU command. The code is now quicker than if the ADD_INT EFB were to be invoked. However, no runtime error is generated if "IN1" is 32767. In this case, "OUT1" would overrun from 32767 to -32768!

7.5 Online functions of the FBD Function Block language

Online Functions

Introduction

There are two animation modes available in the FBD editor:

- Animation of binary variables and links
- Animation of selected objects

These modes are also available on display of a DFB item (command button **Refine...** in the dialog box **Function block: xxx**).

NOTE: If the animated section is used as a transition section for SFC and the transition (and therefore also the transition section) is not processed, the status **DISABLED** appears in the animated transition section.

Animation of binary variables and links

The animation of binary variables and links is activated with the menu command **Online →Animate Booleans**.

In this mode, the current signal status of binary variables, direct addresses in the 0x and 1x range and binary links is displayed in the Editor window.

Animation of selected objects

The animation of the selected objects is activated with the menu command **Online →Animate selected**.

In this mode, the current signal status of the selected links, variables, multi-element variables and literals are displayed in the Editor window.

NOTE: If all variables/links of the section need to be animated, the whole section can be selected with **CTRL**+**A** and then **Online →Animate selected** (**CTRL**+**W**) all variables and links of the section will be animated.

If a numerical value is selected on an input/output, the name of the variable, its direct address and I/O assignment (if available) and its comment will be displayed in the status bar.

NOTE: The selected objects remain selected even after "Animate selected" has been selected again, in order to keep these for a further reading, and/or to be able to easily modify the list of objects.

Color key

There are 12 different color schemes available for animation. An overview of the color scheme and the meaning of each color can be found in the Online help (Tip: Search the online help for the index reference "Colors").

7.6 Creating a program with the FBD Function Block language

Creating a Program in the FBD Function Block Language

Introduction

The following description contains an example for creating a program in the function block language (FBD). The creation of a program in the function block language is divided into 2 main steps:

Step	Action
1	Creating a Section *(see page 235)*
2	Creating the Logic *(see page 236)*

Creating a Section

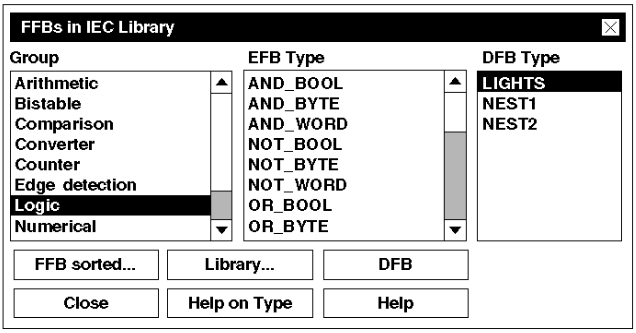
The procedure for creating a section is as follows:

Step	Action
1	Using the **File** →**New Section...** menu command, create a new section and enter a section name. **Note:** The section name (max. 32 characters) is not case-sensitive and must be unique within the whole project. If the name entered already exists, you will be warned and you will have to choose a different name. The section name must comply with the IEC name conventions, otherwise an error message appears. **Note:**In compliance with IEC1131-3 only letters are permitted as the first character of names. However, if you wish to use numbers as the first character, you can enable this using the **Options** →**Preferences** →**IEC Extensions...** → **Allow Leading Digits in Identifiers** menu command.

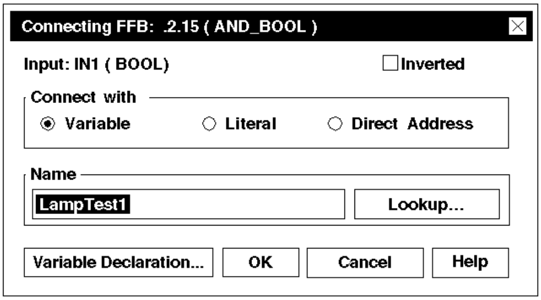
Creating the Logic

The procedure for creating the logic is as follows:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>To insert an FFB into the section, select the <b>Objects →Select FFB...</b> menu command.
<b>Response:</b> The FFB dialog box from the library is opened.



</td></tr>
<tr><td>2</td><td>In this dialog box you can select a library and an FFB from it by using the <b>Library...</b> command button. You can, however, also display the DFBs that you created and select one of them using the <b>DFB</b> command button.</td></tr>
<tr><td>3</td><td>Place the selected FFB in the section.</td></tr>
<tr><td>4</td><td>When all FFBs have been placed, close the dialog box with <b>Close</b>.</td></tr>
<tr><td>5</td><td>Activate the selection mode with <b>Objects →Select Mode</b>, click on the FFB and move the FFBs to the desired position.</td></tr>
<tr><td>6</td><td>Activate the link mode with <b>Objects →Link</b> and connect the FFBs.</td></tr>
<tr><td>7</td><td>Then re-activate select mode with <b>Objects →Select Mode</b> and double-click on one of the unconnected inputs/outputs.
<b>Response:</b> The <b>Connect FFB</b> dialog box opens, where an actual parameter can be allocated to the input/output.



Connecting FFB: .2.15 (AND_BOOL)

Input: IN1 (BOOL) ☐ Inverted

Connect with: ● Variable ○ Literal ○ Direct Address

Name: LampTest1

</td></tr>
</table>

Step	Action
8	Depending on the program logic you can allocate the following to the input/output: ● **Variable** ● Located variable You can allocate a hardware input/output signal to the input/output of the FFB using a located variable. The name of the variable is shown at the input/output in the editor window. ● Unlocated variable You can use the unlocated variable allocated to the input/output of the FFB as a discrete, i.e. when resolving loops, or when transferring values between different sections. The name of the variable is shown at the input/output in the editor window. ● Constant You can allocate a constant to the input of the FFB. The constant can be transferred to other sections. You determine the value of the constant in the variable editor. The name of the constant is shown at the input in the editor window. ● **Literal** You can allocate a literal to the input, i.e. directly allocate a value to the input/output. The value is shown at the input in the editor window. ● **Direct address** You can allocate a hardware input/output signal to the input/output using an address. The address is shown at the input/output in the editor window. **Note:** For an example for invocation of multi element variables see *Calling Derived Data Types, page 588*. **Note:** Unconnected FFB inputs are specified as "0" by default.
9	Save the FBD section with the menu command **File** →**Save Project** .

Ladder Diagram LD

8

Overview

This Chapter describes the Ladder Diagram LD which conforms to IEC 1131.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
8.1	General information about Ladder Diagram LD	240
8.2	Objects in Ladder Diagram LD	242
8.3	Working with the LD Ladder Diagram	257
8.4	Code generation with LD Ladder Diagram	264
8.5	Online functions with the LD Ladder Diagram	265
8.6	Creating a program withLD Ladder Diagram	267

8.1 General information about Ladder Diagram LD

General Information about the LD Ladder Diagram Language

Introduction

This section describes the Ladder Diagram (LD) according to IEC 1131-3.

The structure of a LD section corresponds to a rung for relay switching. The window in the LD editor is shaded with a logic grid, on the left side of which there is the so-called left power rail. This left power rail corresponds to the phase (L ladder) of a rung. With LD programming, in the same way as in a rung, only the LD objects (contacts, coils) which are linked to a power supply, that is to say connected with the left power rail, are "processed". The right power rail, which corresponds to the neutral ladder, is not shown optically. However, all coils and FFB outputs are linked with it internally and this creates a power flow.

Objects

The objects of the programming language LD (Ladder Diagram) help to divide a section into a number of:

- Contacts *(see page 243)*,
- Coils *(see page 245)* and
- FFBs (Functions and Function Blocks) *(see page 248)*.

These objects can be linked with each other through:

- Links *(see page 253)* or
- Actual Parameters *(see page 254)*.

Expansive logic can also be positioned in the LD section in the form of macros (related topics *Macros, page 521*).

Theoretically, each section can contain as many FFBs and also as many inputs and outputs as required. It is therefore advisable to subdivide a whole program into logical units, that is to say into different sections.

Comments can be provided for the logic of the section with text objects (related topics *Text object, page 256*).

Processing Sequence

Basically, LD sections are processed from top to bottom and from left to right.

Networks connected to the left power rail are processed from top to bottom.

The processing sequence of objects (contacts, coils, FFBs) is determined by the data flow within a network.

A detailed description can be found under *Execution sequence, page 260*).

Editing with the Keyboard

Normally editing in Concept is performed with the mouse, however it is also possible with the keyboard (related topics *Shortcut keys in the LD-Editor, page 841*).

In order to make editing with the keyboard easier, you can specify the number of columns per section in the CONCEPT.INI *(see page 1114)* file, after which an automatic carriage return should appear when you are expanding a rung. This means that when you reach the last column, the next object is automatically placed in the second column of the next row. Objects on different rows are automatically linked, i.e. the objects are generated within a common rung.

IEC Conformity

For a description of the IEC conformity of the LD programming language see *IEC conformity, page 849*.

8.2 Objects in Ladder Diagram LD

Overview

This section describes the objects in LD Ladder Diagram.

What's in this Section?

This section contains the following topics:

Topic	Page
Contacts	243
Coils	245
Functions and Function Blocks (FFBs)	248
Link	253
Actual Parameters	254
Text object	256

Contacts

At a Glance

A contact is an LD element that transfers a status on the horizontal link to its right side. This status comes from the boolean AND link of the status of the horizontal link on the left side, with the status of the relevant variable/direct address.

A contact does not change the value of the relevant variable/direct address.

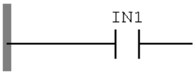
The following contacts are available:

- Closer *(see page 243)*
- Opener *(see page 243)*
- Contact for detection of positive transitions *(see page 243)*
- Contact for detection of negative transitions *(see page 244)*

Closer

On closing, the status of the left link is copied onto the right link, if the status of the relevant boolean variable is ON. Otherwise, the status of the right link is OFF.

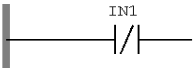
Closer



Opener

On opening, the status of the left link is copied onto the right link, if the status of the relevant boolean variable is OFF. Otherwise, the status of the right link is OFF.

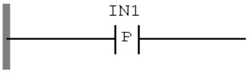
Opener



Contact for detection of positive transitions

With contacts for detection of positive transitions, the right link for a program cycle is ON if a transfer of the relevant boolean variable is made from OFF to ON and the status of the left link is ON at the same time. Otherwise, the status of the right link is OFF.

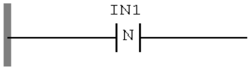
Contact for detection of positive transitions



Contact for detection of negative transitions

With contacts for detection of negative transitions, the right link for a program cycle is ON if a transfer of the relevant boolean variable is made from ON to OFF and the status of the left link is ON at the same time. Otherwise, the status of the right link is OFF.

Contact for detection of negative transitions



Coils

At a Glance

A coil is an LD element which transfers the status of the horizontal link on the left side, unchanged, to the horizontal link on the right side. The status is saved in the relevant variable/direct address.

Start behavior of coils

In the start behavior of PLCs there is a distinction between cold starts and warm starts:

- **Cold start**
  Following a cold start (load the program with **Load online** →**Load**) all variables (independent of type) are set to "0" or, if available, their initial value.
- **Warm start**
  In a warm start (stop and start the program or **Online** → **changes**) different start behaviors are valid for located variables/direct addresses and unlocated variables:
  - **Located variables/direct addresses**
    In a warm start all coils (0x registers) are set to "0" or, if available, their initial value.
  - **Unlocated variable**
    In a warm start all unlocated variables retain their current value (storing behavior).

This different behavior in a warm start leads to particular characteristics in the warm start behavior of LD objects "Coil – set" and "Coil – reset". Warm start behavior is dependent on the variable type used (storing behavior in use of unlocated variables; non storing behavior in use of located variables/direct addresses)

If a buffered coil is required with a located variable or with direct addresses, the RS or SR Function Block from the IEC block library should be used.

Available coils

The following coils are available:

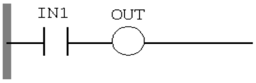
- Coil *(see page 246)*
- Coil - negated *(see page 246)*
- Coil - set *(see page 247)*
- Coil - reset *(see page 247)*
- Coil – positive edge *(see page 246)*
- Coil – negative edge *(see page 246)*

Coil

With coils, the status of the left link is copied onto the relevant Boolean variable and the right link.

Normally, coils follow contacts or EFBs, but they can also be followed by contacts.

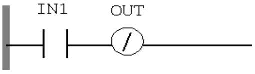
Coil



Coil - negated

With negated coils, the status of the left link is copied onto the right link. The inverted status of the left link is copied onto the relevant Boolean variable. If the left link is OFF, then the right link will also be OFF and the relevant variable will be ON.

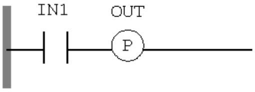
Coil - negated



Coil – positive edge

With coils for detection of positive transfers, the status of the left link is copied onto the right link. The relevant Boolean variable is ON for a program cycle, if a transfer of the left link from OFF to ON is made.

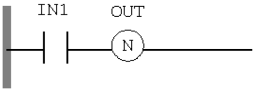
Coil – positive edge



Coil – negative edge

With coils for detection of negative transfers, the status of the left link is copied onto the right link. The relevant Boolean variable is ON for a program cycle, if a transfer of the left link from ON to OFF is made.

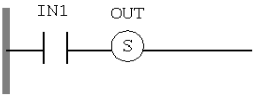
Coil – negative edge



Coil - set

With "set coils", the status of the left link is copied onto the right link. The relevant Boolean variable is set to ON status, if the left link is in ON status, otherwise it remains unchanged. The relevant Boolean variable can only be reset through the "reset coil".

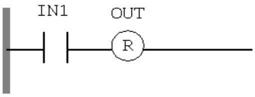
Coil - set



Coil - reset

With "reset coils", the status of the left link is copied onto the right link. The relevant Boolean variable is set to OFF status, if the left link is in ON status, otherwise it remains unchanged. The relevant Boolean variable can only be set through the "set coil".

Coil - reset



Functions and Function Blocks (FFBs)

Introduction

FFB is the generic term for:

- EFB (Elementary Function and Elementary Function Block) *(see page 248)*
- DFB (Derived Function Block) *(see page 250)*
- UDEFB (Derived Elementary Function and Derived Elementary Function Block) *(see page 251)*

EFB

EFB is the generic term for:

- Elementary Function *(see page 248)*
- Elementary Function Block *(see page 249)*

EFBs are functions and function blocks that are available in Concept in the form of libraries. The logic of EFBs is built in C programming language and cannot be changed in the FBD editor.

NOTE: The EFBs AND_BOOL, NOT_BOOL, OR_BOOL, R_TRIG and F_TRIG are not available in LD. Their function is executed with contacts. The MOVE function cannot be used with the data type BOOL.

Elementary Function

Functions have no internal conditions. If the input values are the same, the value at the output is the same for all executions of the function. E.g. the addition of two values gives the same result at every execution.

An Elementary Function is represented graphically as a frame with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The name of the function, that is the function type, is displayed in the center of the frame. The function counter is displayed above the frame.

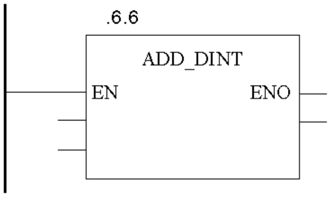
The function counter cannot be changed and always has an .n.m. structure.

.n = current section number

.m = current function number

Functions are only executed if the input EN=1 or if the input EN is grayed out (see also *EN and ENO, page 252*).

Elementary Function



Elementary Function Block

Function Blocks have internal conditions. If the inputs have the same values, the value at the output at every execution is another value. E.g. with a counter, the value on the output is incremented.

A function block is represented graphically as a frame with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The name of the function block, that is the function block type, is displayed in the center of the frame. The instance name is displayed above the frame. The instance name serves as a unique identification for the function block in a project.

The instance name is produced automatically with the following structure: FBI_n_m

FBI = Function Block Instance

n = Section number (current number)

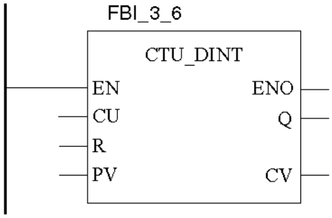
m = Number of the FFB object in the section (current number)

The instance name can be edited in the Properties dialog box of the function block. The instance name must be unique throughout the whole project and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must comply with the IEC name conventions otherwise an error message appears.

NOTE: In compliance with IEC1131-3 only letters are permitted as the first character of instance names. Should numbers be required as the first character however, the **Options →Preferences →IEC Extensions... →Permit Leading Figures in Identifiers** menu command will enable this.

Function blocks are only executed if the input EN=1 or if the input EN is grayed out (see also *EN and ENO, page 252*).

Elementary Function Block



DFB

Derived Function Blocks are function blocks that have been defined in Concept DFB.

With DFBs, there is no distinction between functions and function blocks. They are always treated as function blocks regardless of their internal structure.

A DFB is represented graphically as a frame with double vertical lines and with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The DFB name is displayed centrally within the frame. The instance name is displayed above the frame. The instance name serves as a unique identification for the function block in a project.

The instance name is produced automatically with the following structure: FBI_n_m

FBI = Function Block Instance

n = Section number (current number)

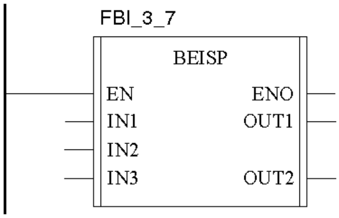
m = Number of the FFB object in the section (current number)

The instance name can be edited in the Properties dialog box of the DFB. The instance name must be unique throughout the whole project and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must comply with the IEC name conventions otherwise an error message appears.

NOTE: In compliance with IEC1131-3 only letters are permitted as the first character of instance names. Should numbers be required as the first character however, the **Options →Preferences →IEC Extensions... →Permit Leading Figures in Identifiers** menu command will enable this.

Derived Function Blocks are only executed if the input EN=1 or if the input EN is grayed out (see also *EN and ENO, page 252*).

Derived Function Block



UDEFB

UDEFB is the generic term for:

- User-defined Elementary Function
- User-defined Elementary Function Block

UDEFBs are functions and function blocks that have been programmed with Concept EFB in C++ programming language and are available in Concept in the form of libraries.

In Concept, there is no functional difference between UDEFBs and EFBs.

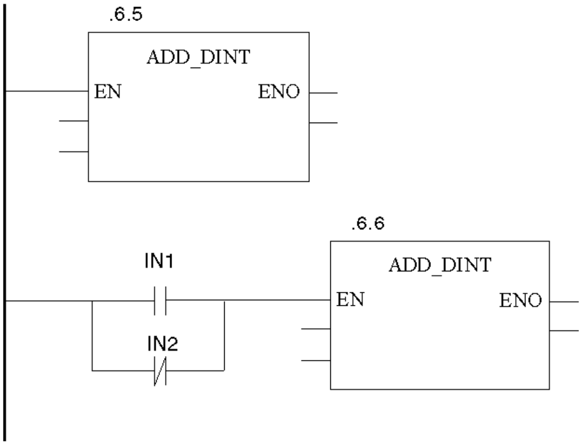
Editing FFBs

FFBs are only edited if at least one Boolean input is linked with the left power rail. If the FFB has no Boolean input, the EN input of the FFB must be used. If the FFB is to be conditionally executed, the Boolean input can be pre-linked through contacts or other FFBs.

NOTE: If the EN input is not linked with the left power rail, it must be deactivated in the Properties dialog box, otherwise the FFB will never be edited.

NOTE: Each FFB without Boolean link to the left power rail gives rise to an error message when downloading onto the PLC.

Connection to an FFB with the left power rail:



EN and ENO

With all FFBs, an EN input and an ENO output can be configured.

EN and ENO configuration is switched on or off in the FFB properties dialog box. The dialog box can be invoked with the **Objects →Properties...** menu command or by double-clicking on the FFB.

If the value of EN is equal to "0" when the FFB is invoked, the algorithms that are defined by the FFB will not be executed and all outputs keep their previous values. The value of ENO is automatically set to "0" in this case.

If the value of EN is equal to "1", when the FFB is invoked, the algorithms which are defined by the FFD will be executed. After successful execution of these algorithms, the value of ENO is automatically set to "1". If an error occurs during execution of these algorithms, ENO will be set to "0".

NOTE: If the EN input is not linked with the left power rail, it must be deactivated in the Properties dialog box, otherwise the FFB will never be edited.

The output behavior of the FFBs does not depend on whether the FFBs are invoked without EN/ENO or with EN=1.

Link

Description

Links are connections between contacts, coils and FFBs.

Several links can be connected with one contact, one coil or one FFB output. The link points are identified with a filled circle.

NOTE: Unconnected contacts, coils and FFB inputs are specified as "0" by default.

Data Types

The data types of the inputs/outputs to be linked must be the same.

Editing Links

Links can be edited in select mode. An overlap with other objects is permitted.

Configuring Loops

No loop can be configured with links because in this case, the execution order in the section cannot be determined uniquely. Loops must be resolved with actual parameters (related topics *Configuring Loops, page 230*).

Horizontal Links

Contacts and coils are automatically connected during positioning with a neighboring, unconnected contact/coil that has the same vertical position. A connection to the power rail is only established if the contact is placed nearby (also see *Defining the Contact Connection, page 1114* in the *Concept INI-File*chapter). If a coil or a contact is positioned on an existing horizontal link, the link is automatically separated and the contact/coil is inserted. When positioned, actual parameters may overlap another object, but they must not go outside the limits of the section frame. If a link to another object is established, this link is checked. If this link is not permitted, you will receive a message and the link will not be generated.

Once objects are positioned, horizontal links with directly adjacent objects are automatically created.

Vertical Links

An exceptional link is the "vertical link". The vertical link serves as a logical OR. With this form of the OR link, 32 inputs (contacts) and 64 outputs (coils, links) are possible.

Actual Parameters

Possible Actual Parameters

In the program runtime, the values from the process or from other actual parameters are transferred to the FFB via the actual parameters and then re-emitted after processing.

Table of possible actual parameters

Element	Actual Parameters
Contacts	● Direct addresses *(see page 67)* ● Located variables *(see page 64)* ● Unlocated variable *(see page 64)*
Coils	● Direct addresses *(see page 67)* ● Located variables *(see page 64)* ● Unlocated variable *(see page 64)*
FFB inputs	● Direct addresses *(see page 67)* ● Located variables *(see page 64)* ● Unlocated variable *(see page 64)* ● Constant *(see page 66)* ● Literals *(see page 66)*
FFB outputs	● Direct addresses *(see page 67)* ● Located variables *(see page 64)* ● Unlocated variable *(see page 64)*

Direct Addresses

The information on/display of direct addresses can be given in various formats. The display format is set in the **Options** →**Preferences** →**Common** dialog box. Setting the display format has no impact on the entry format, i.e. direct addresses can be entered in any format.

The following address formats are possible:

- **Standard Format (400001)**
  The five figure address comes directly after the first digit (the reference).
- **Separator Format (4:00001)**
  The first digit (the reference) is separated from the five figure address that follows by a colon (:).
- **Compact format (4:1)**
  The first digit (the Reference) is separated from the address that follows by a colon (:) where the leading zeros are not specified.

- **IEC Format (QW1)**
  There is an IEC type designation in initial position, followed by the five-character address.
  - %0x12345 = %Q12345
  - %1x12345 = %I12345
  - %3x12345 = %IW12345
  - %4x12345 = %QW12345

Data Types

The data type of the actual parameter must be of BOOL type with contacts and coils. With FFB inputs/outputs, the data type of the actual parameter must match the data type of the inputs/outputs. The only exceptions are generic FFB inputs/outputs, whose data type is determined by the formal parameter. If all actual parameters consist of literals, a suitable data type is selected for the function block.

Initial Values

FFBs, which use actual parameters on the inputs and coils that have not yet received a value assignment, work with the initial values of these actual parameters.

Unconnected Inputs

NOTE: Unconnected contacts, coils and FFB inputs/outputs are specified as "0" by default.

Text object

At a Glance

Text can be positioned in the form of text objects in the Ladder Diagram (LD). The size of these text objects depends on the length of the text. The size of the object, depending on the size of the text, can be extended vertically and horizontally to fill further grid units. Text objects may not overlap with other objects; however they can overlap with links.

Memory space

Text objects occupy no memory space on the PLC because the text is not downloaded onto the PLC.

8.3 Working with the LD Ladder Diagram

Overview

This section describes working with LD Ladder Diagram.

What's in this Section?

This section contains the following topics:

Topic	Page
Positioning Coils, Contacts, Functions and Function Blocks	258
Execution sequence	260
Configuring Loops	262

Positioning Coils, Contacts, Functions and Function Blocks

Positioning Objects

In the LD contact plan editor, the window has a logic grid in the background. The objects are aligned in the bars of this grid (52 x 230 fields) during positioning. With the exception of vertical shorts, FFBs and text fields, all elements require exactly one grid field. Objects can only be positioned within such a field. If an object is positioned between two fields, the object is automatically placed in the nearest field.

When objects are positioned outside the section frame with another object, an error message occurs and the object is not positioned.

When being positioned, contacts and coils are automatically linked with a directly adjacent, unconnected contact/coil, if the contact/ coil has the same vertical position. A link to the power rail is therefore created even if the contact is positioned 2 fields away. If contacts or coils are positioned on existing contacts or coils, the existing ones are replaced by the current ones (only applies to same types, i.e. when replacing coils with coils and contacts with contacts). If a coil or a contact is positioned on an existing horizontal short, the link is automatically separated and the contact/coil is inserted.

When positioned, actual parameters may overlap another object, but they must not go outside the limits of the section frame. If a link to another object is established, this link is checked. If this link is not permitted, you will receive a message and the link will not be generated. When producing links, overlaps and crossings with other links and objects are permitted.

If an FFB is selected, its comment is displayed in the first column of the status line. If an actual parameter is selected, its name and, if applicable, its direct address and its comment are displayed in the first column of the status line.

Automatic Carriage Return

As a keyboard user, you have the possibility of determining the number of columns/fields in the CONCEPT.INI *(see page 1114)* file after which an automatic carriage return will appear during editing as soon as the last column/field is reached. The following object is then inserted into the second column/field and linked to the last object of the previous row. I.e. the objects are created inside the same rung.

Selecting FFBs

Using **Objects →Select FFB...** you can open a dialog for selecting FFBs. This dialog is modeless, which means it is not automatically closed once an FFB has been positioned, but remains open until you close it. If you have several LD sections open and you invoke the dialog, only one dialog box is opened and is available for all sections. The dialog box is not available for any other sections (not LD editor). If the LD sections are changed into symbols (Minimize window), the dialog box is closed. If one of the LD section symbols is invoked again, the dialog box is automatically re-opened.

The first time Concept is started, the FFB is displayed oriented to the library. This means that to select an FFB, the corresponding library must first be selected using the **Library** command button. Then you can select the corresponding group in the **Group** list box. Now, you can select the required FFB from the EFB type list box.

If you do not know which library/group the FFB required is located in, you can invoke an FFB-oriented dialog with the **FFB sorted** command button. This contains all FFBs in all libraries and groups in an alphabetical list.

After each subsequent project start, the view that you select will appear.

Once the FFB has been selected, its position in the section must be selected. The cursor becomes a small FFB and the cross shows the position (upper left corner of the FFB) in which the FFB is placed. The FFB is positioned by clicking on the left-hand mouse button.

Change FFB-Type

With the **Objects →Replace FFBs...** menu command, the FFBs already positioned in the section can be replaced with FFBs of another type (e.g. an AND with an OR). The variables given to the FFB remain if the data type and position of the inputs/outputs are the same in the "old" as the new FFB.

NOTE: FFBs with inputs/outputs of the ANY data type (generic FFBs) cannot be replaced.

Change contact/coil

Contacts and coils which are already positioned can simply be replaced. In order to do this, select the new element and click on the one to be replaced.

Execution sequence

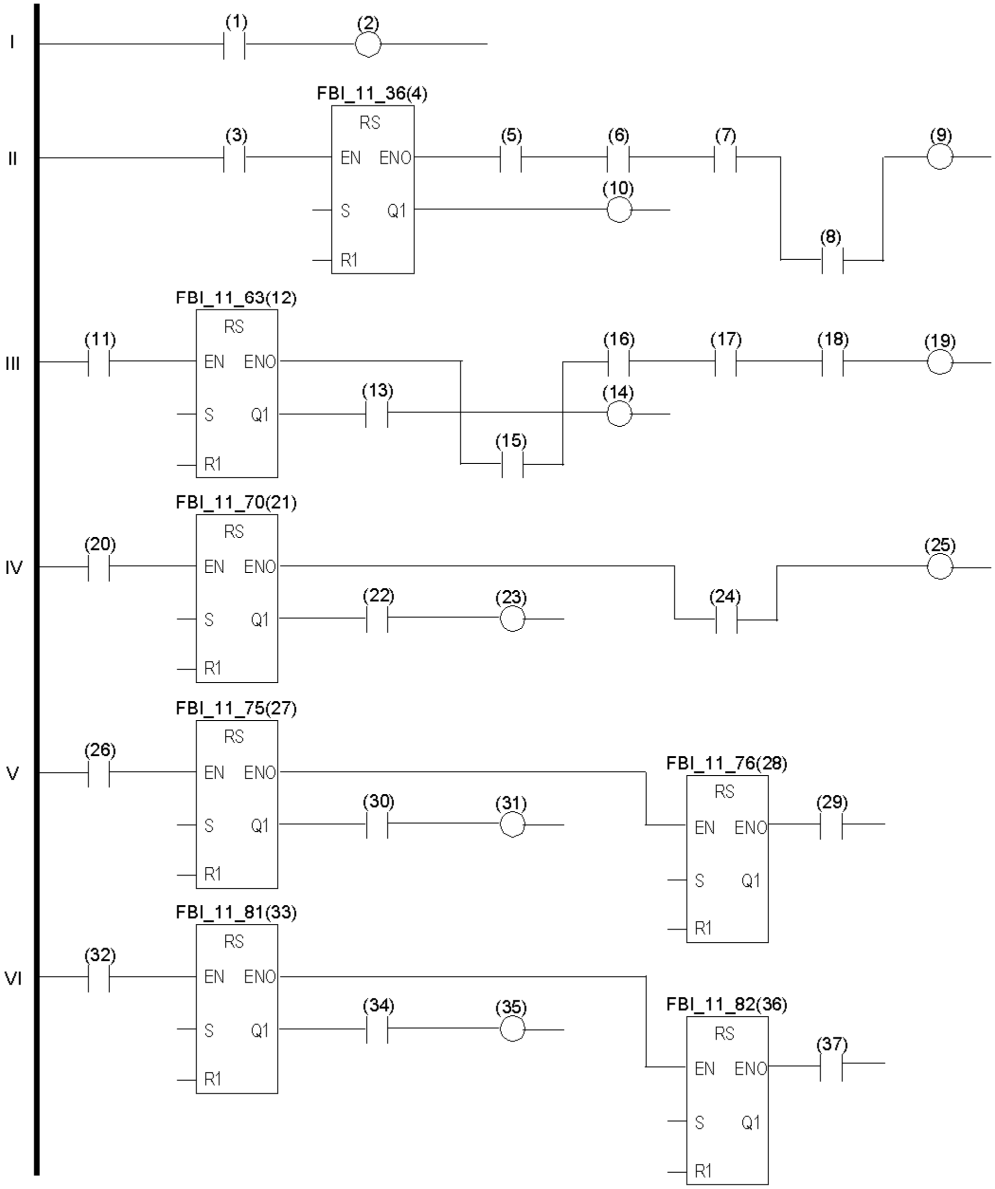
Description

The following applies to the execution sequence in LD sections:

- The execution sequence of networks which are only linked by the left power rail, is determined by the graphic position in which the networks are connected to the left power rail.
  The networks are processed from top to bottom.
  See example below, Networks I-VI).
- The execution sequences of objects (contacts, coils FFBs) are determined by the data flow within a network. This means that the coils and FFBs whose inputs have already received value assignments will be processed first.
- Current paths that begin at outputs (Pins) from FFBs are processed according to the vertical, graphical position of its first object (from top to bottom).
  See example below, Network III):
  Processing after the FFB (FBI_11_63) begins with the current path whose first object is located at the uppermost vertical position (13) and thus follows current path (13)->(14).
  When current path (13)->(14) has been processed, processing of the next current path (15)->(19) begins.
- If the first objects of 2 current paths that begin at outputs (pins) of FFBs, at the same height, the first current path to be processed is that of the object that is farther left.
  See example below, Network IV): (22)->(23) then (24)->(25).
- The position of an FFB is determined by the upper left corner of the FFB.
  See example below.
  Network V: Upper left corner of FFB (FBI_11_76) above contact (30). Process: (28)->(29) then (30)->(31).
  Network VI: Upper left corner of FFB (FBI_11_82) same height as contact (34). Process: (34)->(35) then (36)->(37).

Example

LD section

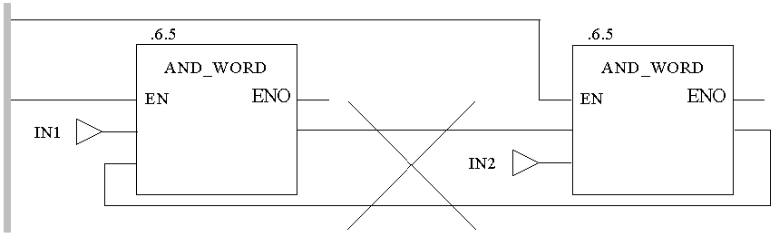


Configuring Loops

Non-permitted Loops

Configuring loops exclusively via links is not permitted, as it is not possible to make a unique specification of the data flow (the output of one FFB is the input of the next FFB, and the output of this one is the input of the first).

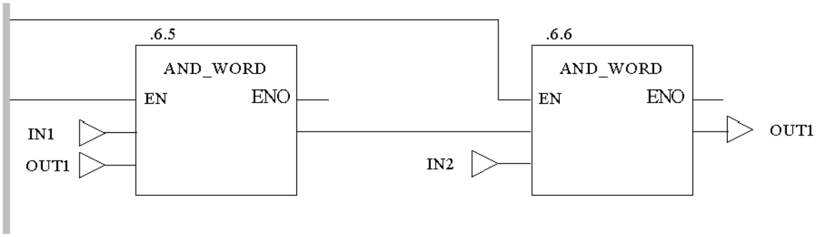
Non-permitted Loops via Links



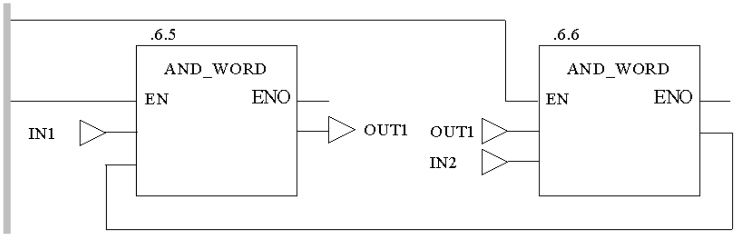
Resolution using an Actual Parameter

This type of logic must be resolved using actual parameters so that the data flow can be determined uniquely.

Resolved loop using an actual parameter: Variant 1



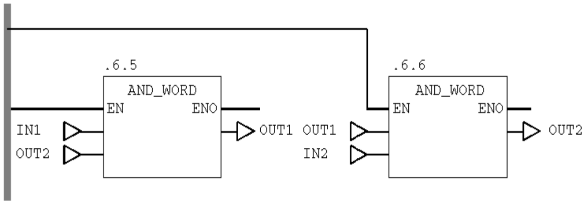
Resolved loop using an actual parameter: Variant 2



Resolution using Several Actual Parameters

Loops using several actual parameters are also allowed.

Loop using several actual parameters



8.4 Code generation with LD Ladder Diagram

Code Generation Options

Introduction

Using the **Project →Code Generation Options** menu command, you can define options for code generation.

Include Diagnosis Information

If you check the **Include Diagnosis Information** check box, additional information for the process diagnosis (e.g. transition diagnosis, diagnosis codes for diagnosis function blocks with extended diagnosis, such as XACT, XLOCK etc.) will be created during code generation. This process diagnosis can be evaluated with MonitorPro or FactoryLink, for example.

Fastest Code (Restricted Checking)

If you check the **Fastest code (Restricted Checking)** check box, a runtime-optimized code is generated. This runtime optimization is achieved by realizing the integer arithmetic (e.g. "+" or "-") using simple CPU commands instead of EFB invocations.

CPU commands are much quicker than EFB invocations, but they do not generate any error messages, such as, for example, arithmetic or array overflow. This option should only be used when you have ensured that the program is free of arithmetic errors.

If **Fastest Code (Restricted Checking)** was selected, the addition IN1 + 1 is solved with the "add" CPU command. The code is now quicker than if the ADD_INT EFB were to be invoked. However, no runtime error is generated if "IN1" is 32767. In this case, "OUT1" would overrun from 32767 to -32768!

8.5 Online functions with the LD Ladder Diagram

Online Functions

Introduction

There are two animation modes available in the LD editor:

- Animation of binary variables and links
- Animation of selected objects

These modes are also available when a DFB instance is displayed (command button **Refine...** in the **Function Block: xxx** dialog box).

NOTE: If the animated section is used as a transition section for SFC and the transition (and therefore also the transition section) is not processed, the status **DISABLED** appears in the animated transition section.

Animation of Binary Variables and Links

The animation of binary variables and links is activated using the **Online →Animate Booleans** menu command.

In this mode, the current signal status of binary variables, direct addresses in the 0x and 1x range and binary links is displayed in the editor window.

Meaning of Colors

Color	Meaning
Contact, coil, input/output, link red	Contact, coil, input/output, link transferring the value 0
Left power rail, contact, coil, input/output, link green	Left power rail, contact, coil, input/output, link transferring the value 1
Variable highlighted in beige	Variable forced
Variable highlighted in purple	Variable cyclically set
The name of the multi-element variable (e.g. motor) highlighted in color.	In the editor, a multi-element variable (e.g. motor) is displayed, in which one or more elements is forced or cyclically set.
The whole element name of the multi-element variable (e.g. right.motor.on) is highlighted in color.	In the editor, an element of a multi-element variable (e.g. right motor on) that is forced or cyclically set is displayed.
The name of the multi-element variable (e.g. right.motor.on) is highlighted in color, but the name of the element is not.	In the editor, an element of a multi-element variable (e.g. right motor on) that is not forced or cyclically set is displayed, but a different element of this multi-element variable is cyclically set or forced.

Animation of Selected Objects

The animation of the selected objects is activated with the **Online** →**Animate Selection** menu command.

In this mode, the current signal status of the selected links, variables, multi-element variables and literals is displayed in the editor window.

NOTE: If you want to animate all variables/links in the section, you can select the whole section using **CTRL**+**A** and then animate all variables and links in the section using **Online** →**Animate Selection** (**CTRL**+**W**).

If a numerical value is selected on an input/output, the name of the variable, its direct address and I/O mapping (if existent) and its comment will be displayed in the status bar.

NOTE: The selected objects remain selected even after "animate selection" has been selected again, to retain these objects for a further reading, and/or to be able to easily modify the list of objects.

Color key

There are 12 different color schemes available for animation. An overview of the color scheme and the meaning of each color can be found in the Online help Tip: Search the online hlep for the index reference "Colors").

8.6 Creating a program withLD Ladder Diagram

Creating a Program in LD

Introduction

The following description contains an example for creating a program in Ladder Diagram (LD). The creation of a program in LD Ladder Diagram is divided into 2 main steps:

Step	Action
1	Creating a Section *(see page 267)*
2	Creating the Logic *(see page 268)*

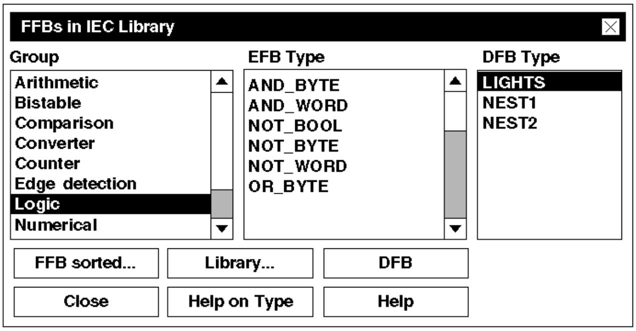
Creating a Section

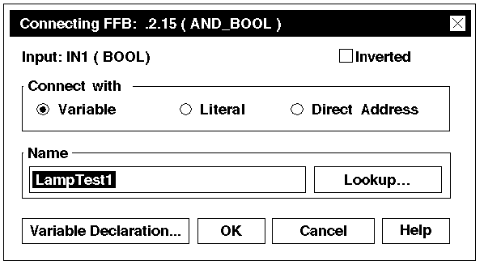
The procedure for creating a section is as follows:

Step	Action
1	Using the **File** →**New Section...** menu command, create a new section and enter a section name. **Note:** The section name (max. 32 characters) is not case-sensitive and must be unique within the whole project. If the name entered already exists, you will be warned and you will have to choose a different name. The section name must comply with the IEC name conventions, otherwise an error message appears. **Note:**In compliance with IEC1131-3 only letters are permitted as the first character of names. However, if you wish to use numbers as the first character, you can enable this using the **Options** →**Preferences** →**IEC Extensions...** → **Allow Leading Digits in Identifiers** menu command.

Creating the Logic

The procedure for creating the logic is as follows:

Step	Action
1	To insert a contact or coil in the section, open the **Objects** main menu and select the desired contact or coil. Contacts and coils can also be selected using the tool bar. Place the contact or coil in the section.
2	To insert an FFB into the section, select the **Objects** →**Select FFB...** menu command. **Response:** The **FFBs from Library** dialog box is opened.  FFBs in IEC Library Group Arithmetic Bistable Comparison Converter Counter Edge detection Logic Numerical EFB Type AND_BYTE AND_WORD NOT_BOOL NOT_BYTE NOT_WORD OR_BYTE DFB Type LIGHTS NEST1 NEST2 FFB sorted... Library... DFB Close Help on Type Help
3	In this dialog box you can select a library and an FFB from it by using the **Library...** command button. You can, however, also display the DFBs that you created and select one of them using the **DFB** command button.
4	Place the selected FFB in the section.
5	When all FFBs have been placed, close the dialog box with **Close**.
6	Activate select mode using **Objects** →**Select Mode**, and move the contacts, coils and FFBs to the required position.
7	Activate link mode with **Objects** →**Link**, and connect the contacts, coils and FFBs. Connect the contacts, FFBs and the left power rail.
8	Then re-activate select mode with **Objects** →**Select mode**, and double-click on a contact or coil. **Response:** The **Properties: LD objects** dialog box is opened, in which you can allocate an actual parameter to the contact/coil.

Step	Action
9	Depending on the program logic you can allocate the following to the contact/coil: ● **Variable** ● **Located variable** You can allocate a hardware input/output signal to the input/output using a located variable. The name of the variable is shown at the input/output in the editor window ● **Unlocated variable** You can use the unlocated variable allocated to the input/output as a discrete, i.e. to resolve loops, or to transfer values between different sections. The name of the variable is shown at the input/output in the editor window. ● **Direct address** You can allocate a hardware input/output signal to the input/output using an address. The address is shown at the input/output in the editor window. **Note:** For an example for invocation of multi-element variables see *Calling Derived Data Types, page 588*. **Note:** Unconnected FFB inputs are specified as "0" by default.
10	To connect the FFB input/outputs to the actual parameters, double-click on one of the unconnected input/outputs. **Response:** The Connect FFB dialog box is opened, in which you can allocate an actual parameter to the input/output.  Connecting FFB: .2.15 (AND_BOOL) Input: IN1 (BOOL) □ Inverted Connect with ● Variable ○ Literal ○ Direct Address Name LampTest1 Lookup... Variable Declaration... OK Cancel Help

Step	Action
11	Depending on the program logic you can allocate the following to the input/output: ● **Variable** ● Located variable You can allocate a hardware input/output signal to the input/output using a located variable. The name of the variable is shown at the input/output in the editor window ● Unlocated variable You can use the unlocated variable allocated to the input/output as a discrete, i.e. to resolve loops, or to transfer values between different sections. The name of the variable is shown at the input/output in the editor window. ● Constant You can allocate a constant to the input. The constant can be transferred to other sections. You determine the value of the constant in the variable editor. The name of the constant is shown at the input in the editor window. ● **Literal** You can allocate a literal to the input, i.e. directly allocate a value to the input/output. The value is shown at the input in the editor window. ● **Direct address** You can allocate a hardware input/output signal to the input/output using an address. The address is shown at the input/output in the editor window. **Note:** For an example for invocation of multi-element variables see *Calling Derived Data Types, page 588*. **Note:** Unconnected FFB inputs are specified as "0" by default.
12	Save the LD section using the **File** →**Save Project** menu command.

Sequence language SFC

9

Overview

This Chapter describes the sequence language SFC which conforms to IEC 1131.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
9.1	General information about SFC sequence language	272
9.2	SFC sequence language elements	274
9.3	Working with the SFC Sequence Language	293
9.4	Online functions of the SFC sequence language	310

9.1 General information about SFC sequence language

General information about SFC language

At a Glance

The sequence language SFC is described in this section according to IEC 1131-3.

In the SFC (Sequential Function Chart) sequence language, a section is split into single configured sequential steps, through steps and transitions, which alternate in the sequence plan.

Objects

A sequential control uses the following objects when creating a program:

- Step *(see page 275)*
- Transition *(see page 280)*
- Jump *(see page 285)*
- Connection *(see page 284)*
- Alternative branch *(see page 287)*
- Simultaneous branch *(see page 290)*
- Alternative connection *(see page 289)*
- Parallel connection *(see page 291)*
- Text object *(see page 292)*

Structure of an SFC section

Steps and transitions are linked with one another through directional links. Two steps can never be directly linked, and must always be separated by a transition. The processes of the active signal status take place along the directional links, triggered by the connecting of a transition. The direction of the string process follows the directional links and runs from the under side of the predecessor step to the top side of the successive step. Branches are processed from left to right.

A jump can be put in the place of a step. Step strings are always concluded with a jump to another step on the same step string. It is run down cyclically.

Nil or more action belong to every step. Steps without action are known as waiting steps. A condition for transition belongs to every transition.

Editing with the keyboard

Normally editing in Concept is performed with the mouse, however it is also possible with the keyboard (see also *Short Cut Keys in the FBD and SFC Editor, page 837*)

IEC conformity

For a description of the IEC conformity of the SFC programming language see *IEC conformity, page 849*.

9.2 SFC sequence language elements

Overview

This section describes the SFC sequence language elements.

What's in this Section?

This section contains the following topics:

Topic	Page
Step	275
Action	278
Transition	280
Transition section	282
Link	284
Jump	285
Alternative Branch	287
Alternative connection	289
Parallel branch	290
Parallel connection	291
Text object	292

Step

Introduction

A step is represented using a block that contains a step name. Step names must be unique within the project.

A step becomes active when the upstream transition is satisfied and is normally inactive when the downstream transition is satisfied.

Initial Step

A special case with steps is the initial step. The initial status of a SFC section is characterized by the initial step, which is active when initializing the project containing the section. A step in a SFC section must always be defined as an initial step. In Concept it is possible to define a step in the middle of a step string as initial.

The initial step is denoted by double lined borders.

Waiting Step

Zero or more actions belong to every step. Steps without action are known as waiting steps.

Step Delay Time

A time can be entered, which is the least amount of time the step must be active for. This is called the step delay time (step duration).

NOTE: This time is only applicable to the step, not for the actions allocated to it. Individual times can be defined for these.

Maximum Supervision Time

The maximum supervision time specifies the maximum time in which the step should normally be active. If the step is still active after this period of time, an error message occurs, which you can view using the **Online →Event Viewer**. In animation mode, the error is additionally identified by a colored outline around the step object.

NOTE: This time supervision applies only to the step, not to the actions allocated to it. Individual times can be defined for these.

Minimum Supervision Time

The minimum supervision time sets the minimum time for which the step should normally be active. If the step is still active after this period of time, an error message occurs, which you can view in the **Online** →**Event Viewer**. In animation mode, the error is additionally identified by a colored outline around the step object.

NOTE: This time supervision applies only to the step, not to the actions allocated to it. Individual times can be defined for these.

Coordinating the Times

Step delay time< minimum supervision time< maximum supervision time

Setting the Times

In the properties dialog, the time values can be entered directly as time literals or can be set as multi element variables of data type SFCSTEP_TIMES. The values can be automatically determined in learn supervision time mode.

The time literals can be modified in animation mode.

’SFCSTEP_TIMES’ Variable

In ’SFCSTEP_TIMES’ variable usage, the learned times of these variables are assigned as the initial values. If these initial values are to be used for a long period of time, corresponding elements (min., max.) of these variables must not be written. After the supervision times have been learned, the modified initial values must be downloaded to the PLC using **Online** →**Download Changes**.

The ’SFCSTEP_TIMES’ variable can be used everywhere and has the following structure:

```
’varname’: SFCSTEP_TIMES
   delay: TIME
   min: TIME
   max: TIME
```

The elements have the following meaning:

- ’varname’.delay = delay time
- ’varname’.min = minimum supervision time
- varname’.max = maximum supervision time

Step Variable

Every step is implicitly allocated a (read only) variable of data type SFCSTEP_STATE. This step variable has the name of the allocated step. The step variable can be used everywhere and has the following structure:

```
'Step name': SFCSTEP_STATE
   t: TIME
   x: BOOL
   tminErr: BOOL
   tmaxErr: BOOL
```

The elements have the following meaning:

- ’Step name’.t = current dwell time in step
- ’Step name’.x
  - 1: Step active
  - 0: Step inactive
- ’Step name’.tminErr
  - 1: Underflow of minimum supervision time
  - 0: No underflow of minimum supervision time
- ’Step name’.tmaxErr
  - 1: Overflow of maximum supervision time
  - 0: No overflow of maximum supervision time

Action

At a Glance

The actions, which are to be performed, as the step is active must be connected to the step.

Actions are declared in the properties dialog of the triggering step, see *Declaring actions, page 300*.

A step can be assigned none or several actions. A step which is assigned no action, has a waiting function, i.e. it waits until the assigned transition is completed.

An action is a variable of BOOL data type.

The control of actions is expressed through the use of identifiers.

Signal assignment

The following signals can be assigned to an action:

- **Direct address**
  An action can be assigned a hardware output via a direct address. In this case, the action can be used as an enabling signal for a transition, as an input signal in another section and as an output signal for the hardware.
- **Variable**
  The action can be used as an input signal with assistance from a variable in another section. This variable is also called action variable.
  - **Unlocated variable**
    With Unlocated variablethe action can be used as an enabling signal for a transition and as an input signal in an FBD section. Unlocated variables are declared in the Variable Editor *(see page 543)*.
  - **Located variable**
    With Located variable the action can be used as an enabling signal for a transition, as an input signal in another section and as an output signal for the hardware. Located variables are declared in the Variable Editor *(see page 543)*.

Direct addresses

The information on/display of direct addresses can be given in various formats. The display format is set in the dialog box **Options** →**Presettings** →**Joint**. Setting the display format has no impact on the entry format, i.e. direct addresses can be entered in any format.

The following address formats are possible:

- **Standard format (X00001)**
  The five-character address comes directly after the first digit (the Reference).
- **Separator format (X:00001)**
  The first digit (the Reference) is separated from the following five-character address by a colon (:).
- **Compact format (X:1)**
  The first digit (the Reference) is separated from the following address by a colon (:), and the leading zeros of the address are not given.
- **IEC format (XW1)**
  In first place, there is an IEC identifier, followed by the five-character address.
  - %0x12345 = %Q12345
  - %1x12345 = %I12345
  - %3x12345 = %IW12345
  - %4x12345 = %QW12345

Transition

Introduction

A transition specifies the condition through which the check of one or more pre-transition steps passes on to one or more consecutive steps along the corresponding link.

Transition Condition

A transition condition is one of the variables of data type BOOL allocated to the transition.

Transition conditions are declared in the properties dialog of the transition, see also *Declaring a Transition, page 305*.

The transition condition can be:

- a direct address (input or output),
- a variable (input or output) or
- a Transition Section *(see page 282)*.

Variable name position:

If...	Then...
If you allocate a direct address or a variable to the transition.	Then the name of the address/variable is displayed below the transition icon.
If you allocate a transition section to the transition.	Then the name of the transition section is displayed above the transition icon.

NOTE: The variable or address allocated to the transition is only read by the transition, never written.

Enabling a Transition

A transition is enabled if the steps immediately preceding it are active. Transitions whose immediately preceding steps are not active are not analyzed.

NOTE: If no transition condition is defined, the transition will never be active.

Transition Switch Time

The transition switch time can theoretically be as short as possible, but can never be zero. The transition switch time lasts at least the duration of the scan.

Transition Diagnosis

Transition switching can be supervised by the Transition Diagnosis *(see page 319)*.

Transition Trigger Sweep

Transition trigger sweep occurs when the transition is enabled and the associated transition conditions are satisfied.

Triggering a transition leads to the disabling (resetting) of all immediately preceding steps that are linked to the transition, followed by the activation of all immediately following steps.

If triggering a transition leads to the activation of several steps at the same time, then the sequence belonging to these steps is called Parallel Chain *(see page 290)*. After simultaneous activation, each of these chains is processed independently of each other. To emphasize this specific type of construction, the branch and connection of parallel chains are displayed with a double horizontal line.

Transition section

At a Glance

For every Transition *(see page 280)* a transition section can be created. This is a section containing the logic of the transition condition and it is automatically linked with the transition.

Generating a transition section

Transition sections are generated in the properties dialog of the transition, see also *Declaring a Transition, page 305*.

Name of transition section

Name of transition section:

If…	Then…
If in the dialog **Options** →**Preferences** → **Graphical Editors...** the option **Dynamically enumerated** has been selected.	Then the alias designation of the transition is displayed in the **Transition properties** dialog automatically.
Should a name for the transition section be entered manually.	Please ensure that the name is unique throughout the whole project (the name is not case-sensitive). If the section name entered already exists, a warning is given, and another name must be chosen. The name must correspond to the IEC Name conventions, otherwise an error message appears.

NOTE: Do NOT alter the name of a transition section through **Data file** → **Section properties**, otherwise the link to the transition is will be lost.

Occupying a transition section

When first opening the transition section (**Edit...** key in the **Transition properties** dialog) this is automatically generated. The name of the transition section is displayed above the transition symbol in the SFC editor.

Altering the transition conditions

Should another option be selected after the creation of the transition section as **Transition section**, a query appears, whether the transition section should be deleted. If the question is replied in the negative, the transition section remains.

A list can be displayed with the currently unused transition section with help from the command button **Look up...** .

Programming languages for transition section

FBD, LD, IL and ST are possible as programming languages for transition sections.

The programming language to be used can be defined in the dialog **Options** → **Preferences...** →**Common...** with the option **Language for transition sections**. Should the FBD programming language be selected, the section is automatically preallocated with a UND block with 2 inputs whose outputs is preallocated with the name of the transition section. The proposed block can then be linked or altered. No such provision is evident for the other programming languages.

Editing function for transition section

The editing function for transition sections is restricted as opposed to "normal" sections in the following ways:

- The transition section only has one single output (transition variable), whose data type is BOOL. The name of this variable must be identical to the name entered in the **Transition section** field.
- The transition variable can only be used once in written form.
- Only functions can be used, Function Blocks cannot.
- There is only one network, i.e. all functions used are linked with each other either directly or indirectly.
- Transition sections can only be reached via the menu command button **Edit...** in the **Transition properties** dialog. They do not appear in the **Open section** dialog.
- In the **Delete section** dialog transition sections are denoted by a "T" in front of the section name.

Transition section animation

If the transition, and therefore the transition section, is not processed, the status INHIBITED appears in the animated transition section.

Link

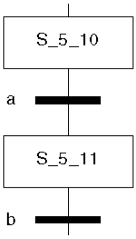
At a Glance

Links connect steps and transitions. Links are normally generated automatically when positioning objects. If objects are positioned in cells which do not immediately follow each other, a link must explicitly be made.

Simple sequences

The change of step and transition is consequentially repeated with simple sequences.

A process of S_5_10 to S_5_11 only takes place, if step 5_10 is in an active state and the condition for transition a is true.



Jump

General information

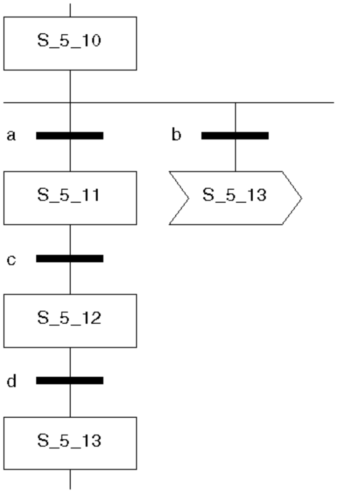
A jump enables a program to continue in another place. Jumps into a Parallel chain *(see page 290)* in or out of a parallel chain are not possible.

Differences are made between chain jumps and chain loops with jumps.

Chain jump

A chain jump is a special case of alternative branch, with one or more branches containing no steps.

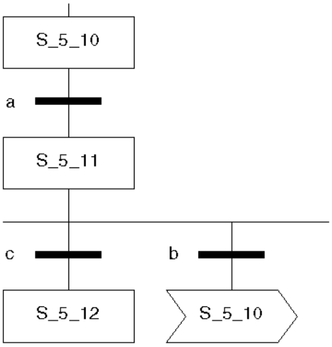
A process of S_5_10 via S_5_11 and S_5_12 after S_5_13 only occurs, if S_5_10 is active and the condition for transition a is true. A process of S_5_10 directly after S_5_13 only occurs, if S_5_10 is active and the condition for transition b is true and a is false.



Chain loop

A chain loop is a special case of alternative branch, with which one or more branches lead back to a previous step.

A process of S_5_11 via S_5_10 only occurs if the condition for transition c is false and b is true.



Alternative Branch

Introduction

The alternative branch offers the possibility to program branches conditionally in the control flow of the SFC structure.

Structure

With alternative branches, as many transitions follow a step under the horizontal line as there are different sequences. Only one of these transitions can ever be switched. The branch to be solved is determined by the result of the transition conditions of the transitions, which come after the alternative branch.

Processing Sequence

Branch transitions are processed from left to right. If a transition condition is satisfied, the remaining transitions are no longer processed The branch with the satisfied transition is activated. This gives rise to a left to right priority for branches.

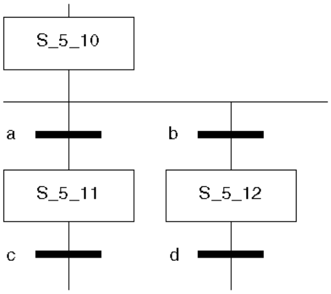
If none of the transitions is switched, the currently set step remains set.

Processing

Sequence processing:

If...	Then...
If S_5_10 is active and the transition condition a is true.	Then a sequence from S_5_10 to S_5_11 occurs.
If S_5_10 is active and the transition condition b is true and a is false.	Then a sequence from S_5_10 to S_5_12 occurs.

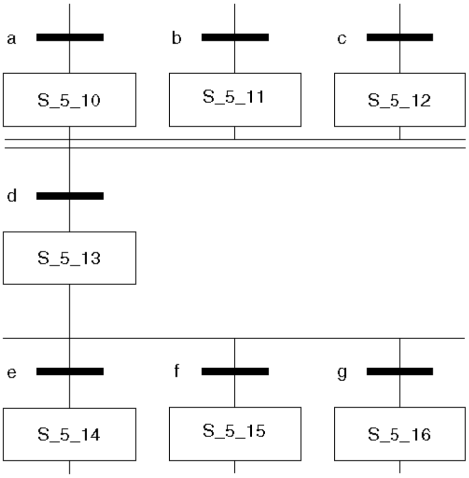
Sequence processing:



Alternative Branch after Parallel Joint

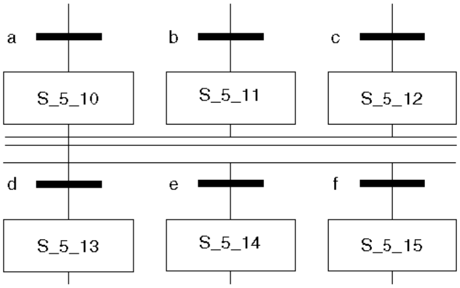
According to IEC 1131-3, alternative branches may not directly follow parallel joints. The joint and the branch must be separated by a transition step sequence.

Example:



If you want to insert an alternative branch directly after a parallel joint, you can use the **Options** →**Preferences** →**Graphic Editors** →**Allow Alternative Branches after Parallel Joints** to do so.

Example:



Joint

All alternative branches must be rejoined to a single branch through Alternative Joints *(see page 289)* or Jumps *(see page 285)*.

Alternative connection

At a Glance

In the alternative connection, the various branches of an alternative branch are again connected to one branch in which additional processing can be performed. This connection can also be performed with a jump.

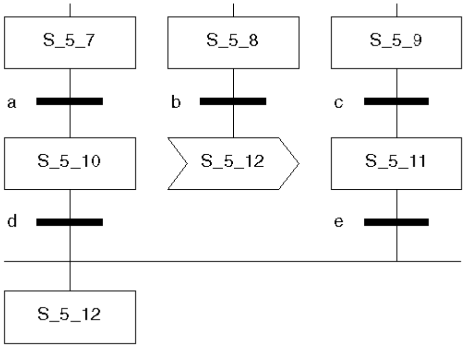
Processing

Sequence processing:

If…	Then…
If S_5_10 is active and the transition condition d is true.	Then a process of S_5_10 to S_5_12 takes place.
If S_5_8 is active and the transition condition b is true, and therefore a jump to S_5_12 is performed.	Then a process of S_5_8 to S_5_12 takes place.
If S_5_11 is active and the transition condition e is true.	Then a process of S_5_11 to S_5_12 takes place.

NOTE: Only a single one of these branches is active, corresponding to the transition condition in the alternative branch.

Sequence processing:



Parallel branch

At a Glance

With parallel branches, the edit is split into two or more strings, which will be processed in parallel Only a joint transition immediately through the horizontal double synchronization lines is possible.

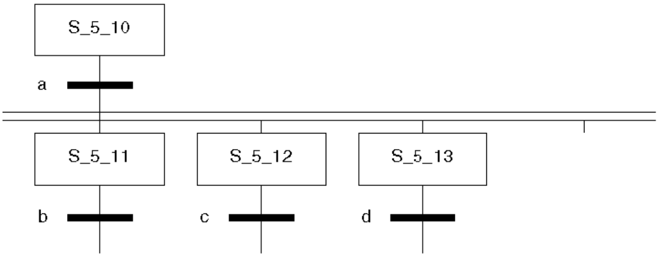
Processing

Processing a sequence:

If…	Then…
If S_5_10 is active and the transition condition a, which shares the same transition, is likewise true.	Then a process of S_5_10 to S_5_11, S_5_12,… takes place.

NOTE: After the simultaneous activation of S_5_11, S_5_12 etc., the sequences run independent of each other.

Processing a sequence:



Definition of initial steps

If a step is to become an initial step within a parallel branch, a step must be defined as the initial step in each branch of the parallel branch.

Parallel connection

At a Glance

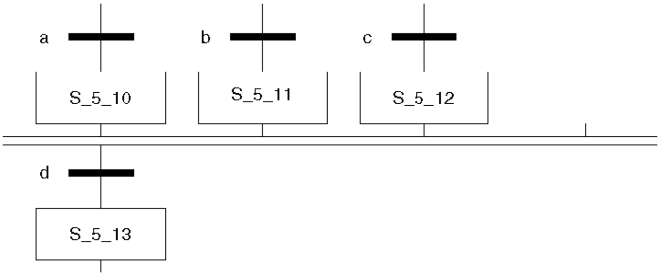
The parallel connection reconnects two or more parallel branches to a branch. The transition to a parallel connection is evaluated when all previous steps of the transition are set. Only a joint transition immediately through the double horizontal synchronisation lines is possible.

Processing

Processing a sequence:

If…	Then…
If S_5_10, S_5_11 etc. are active at the same time and the transition condition d, sharing a joint transition, is true.	Then a process of S_5_10, S_5_11, …to S_5_13 takes place.

Processing a sequence:



Text object

At a Glance

Text can be positioned in the form of text objects using SFC sequence language. The size of these text objects depends on the length of the text. This text object is at least the size of a cell and can be vertically and horizontally enlarged to other cells according to the size of the text. Text objects can only be placed in free cells.

Memory space

Text objects occupy no memory space on the PLC because the text is not downloaded onto the PLC.

9.3 Working with the SFC Sequence Language

Overview

This section describes working with the SFC sequence language.

What's in this Section?

This section contains the following topics:

Topic	Page
General information on editing objects	294
Declaring step properties	297
Declaring actions	299
Identifier	302
Declaring a Transition	305
Alias Designations for Steps and Transitions	307

General information on editing objects

At a Glance

In the SFC editor the background consists of a logical grid. SFC objects can theoretically be placed in every unoccupied cell. If a link with another object is established (explicitly or by vertically placing objects in neighboring cells), this link will be tested. If this link is not permitted, a report of this is given and the object is not inserted.

Steps, transitions and jumps each require a cell. Parallel branches, parallel connections, alternative branches and alternative connections do not require a separate cell each, but are inserted into the corresponding cell of the step or transition.

Maximum number of elements

To prevent step strings being subdivided, 99 linked steps with the transitions are vertically shown along with a locking jump with its transition. To limit the complexity and to enable the animation to be performed, the number of objects (Steps + Transitions + Branches + Connections) in one section is limited to 2000.

Inserting Objects

The SFC object (Step, Transition etc.) can be inserted individually via the menu command in the main menu **Objects** or in the form of a a group (Step transition string, structured parallel string etc.) of the required size.

After selection of the object, a position in the step string can be selected, in which the object should be inserted. If the position selected is already occupied, space is made before insertion into the step string, if desired, and then the object placed in it. If the object is placed on a connection, it is separated, the object is inserted and a link to the newly placed object is generated.

Shifting objects

If the object is shifted onto a connection, it is separated, the object is inserted and a link to the newly placed object is generated.

Copying steps

By copying and inserting it is possible to copy steps through projects. Since the definition of actions displays a reference to a variable, which is defined by the Variable Editor for the particular project, copying between projects can result in this reference no longer being valid. In this instance, the action is deleted, the action list is updated and an error message is displayed.

Deleting steps

Steps can only be deleted after an action has been saved if the action(s) were unconnected before the step was performed.

Selecting an object

The procedure for selecting an object is as follows:

Step	Action
1	With **Objects** →**Selection mode** go to selection mode.
2	Position the cursor on the object to be selected and left-click. **Reaction:** The selected object is displayed in a blue border.

Selecting several objects (by pressing Shift)

The procedure for selecting several objects (by pressing Shift) is as follows:

Step	Action
1	With **Objects** →**Selection mode** go to selection mode.
2	Position the cursor on the object to be selected first and left-click.
3	Press and hold the **Shift** key, select additional objects and left-click. **Reaction:** The selected objects are displayed in a blue border.

Selecting several objects (by using the rubber band function)

The procedure for selecting several objects (by using the rubber band function) is as follows:

Step	Action
1	With **Objects** →**Selection mode** go to selection mode.
2	Press and hold the left mouse button, and pull a border over the objects to be selected. **Reaction:** On releasing the mouse key, all objects touching the border will be selected. The selected objects are displayed in a blue border.

Selecting all objects in a column/line

The procedure for selecting all objects in a column/line is as follows:

Step	Action
1	With **Objects** →**Selection mode** go to selection mode.
2	In the column ruler/line ruler, click on the column number/line number whose objects are to be selected. Note: To select several columns/lines, press and hold the **Shift** key. **Reaction:** The selected objects are displayed in a blue border.

Inserting additional columns

The procedure for inserting additional columns within an existing step string is as follows:

Step	Action
1	With **Objects** →**Selection mode** go to selection mode.
2	In the column ruler, click on the column number in front of which the insertion is to be performed. Note: In order to insert several columns, press the **Shift** key to select several columns and insert a corresponding number of empty spaces.
3	Use the menu command **Edit** →**Insert**. **Reaction:**From the selected column, the entire step string is moved one column to the right. The links (branches) will remain intact.

Inserting additional lines

The procedure for inserting additional lines within an existing step string is as follows:

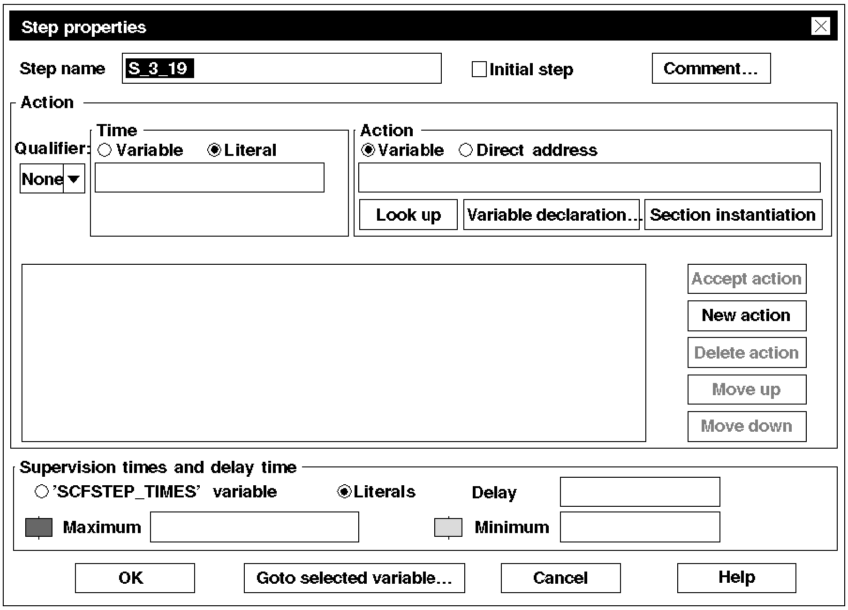
Step	Action
1	With **Objects** →**Selection mode** go to selection mode.
2	In the line ruler, click on the line number in front of which the insertion is to take place. Note: Should the insertion of several lines be required, several lines are selected and a corresponding number of empty spaces are inserted by pressing the **Shift** key.
3	Use the menu command **Edit** →**Insert**. **Reaction:** From the selected line, the entire step string is moved one line downwards. The links (branches) therefore remain even.

Declaring step properties

Introduction

The step properties are declared in the properties dialog of the step.

Declaring step properties:



Step properties

Step name S_3_19

☐ Initial step

Action

Qualifier: None

Time: ○ Variable ● Literal

Action: ● Variable ○ Direct address

Supervision times and delay time

○ 'SCFSTEP_TIMES' variable ● Literals

Delay

▨ Maximum

□ Minimum

Declaring step properties

The following description contains an example of declaring the step properties:

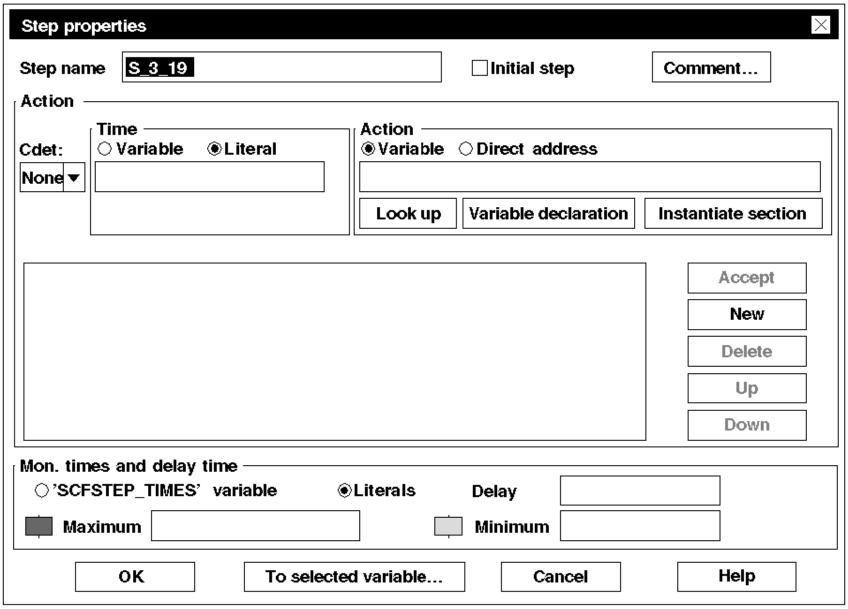
Step	Action
1	With **Objects →Selection mode** go to selection mode.
2	Double-click on a step. **Result:** The dialog **Step properties** of the step opens.
3	A name can be manually defined for the step, or the proposed name can remain. If a name is to be assigned, please note that the step name (max. 32 characters) must be unique for the entire project. If the step name entered already exists, a warning is given and another name must be chosen. The step name must correspond to the IEC name conventions, otherwise an error message is displayed. **Note:** In accordance with IEC1131-3, only letters are permitted as the first character of step names. Should numbers be required as the first character, however, use the menu command **Options →Preferences →IEC Extensions →Allow leading digits in identifiers.** Step names may not end in 4 digits (e.g. xxx_1234). This ending is reserved in case in **Options →Preferences →Graphical Editors...** the options button **Dynamic numbered** is activated. Instead of the free names an alias designation can also be selected, see also *Alias Designations for Steps and Transitions, page 307* This is then shown in SFC and FBD sections and with search functions, application documentation and analysis.
4	Next, define whether or not the step is the initial step of the sequence. A initial step must be defined for each sequence.
5	If desired, the Supervision time and delay time can be defined for the step. The time values can be entered in the properties dialog either directly as time duration literal (this can be automatically transmitted in the Learn Supervision time mode, see also *Learn monitoring times, page 316*) or as multi-element variable of SFCSTEP_TIMES data type, see also *’SFCSTEP_TIMES’ Variable, page 276*. Here: Delay time< minimum Supervision time< maximum Supervision time
6	Using the button **Comment** call up the dialog box **Enter with comment**, in which a comment on the step may be entered. This comment is shown in the status bar of the editor window, when the step is selected.

Declaring actions

At a Glance

The actions are declared in the properties dialog of the step.

Declaring actions:



Step properties

Step name: S_3_19

□ Initial step

Action

Cdet: None

Time: ○ Variable ● Literal

Action: ● Variable ○ Direct address

Mon. times and delay time: ○ 'SCFSTEP_TIMES' variable ● Literals

Declaring actions

The following description contains an example of declaring the actions:

Step	Action
1	With **Objects** →**Selection mode** go to selection mode.
2	Double-click on a step. **Reaction:** The dialog **Step properties** of the step is opened.
3	From the Cdet list, select an Identifier *(see page 302)* for the Action. In this way, the behavior of the action is determined (e.g. saving, non saving, delayed etc.). **Note:** With the identifiers L, D, and DS, in the text box **Time duration:** an additional time duration of TIME data type must be defined.
4	Next define the type of action (variable or dirct address) in the zone **Type:** with the option buttons.
5	● If the **Variable** has been selected, it is possible with the button **Var. declaration...**to open the Variable Editor and define a new output variable there. Also with the command button **Look up...** a list of all the variables can be shown and one selected through Select. ● If the **Direct address** has been selected, in the text box **Direct address:** the output address must be entered.
6	After all the definitions for the actions have been met, confirm this with the command button **New** **Note:** Confirmation with the **Enter** key is not possible in this case and leads to an error message

Altering an action

The procedure for altering an action declaration is as follows:

Step	Action
1	With **Objects** →**Selection mode** go to selection mode.
2	Double-click on a step. **Reaction:** The dialog **Step properties** of the step is opened.
3	To alter an action declaration, select an action in the list. **Reaction:** All definitions (identifiers, time duration, variable or address and type) of the action are transferred into the corresponding text boxes and lists.
4	If these definitions are altered, as described in the *Declaring actions, page 300* section.
5	● Should it be necessary to assign these new definitions as a new action in the step, use the command button **New**. **Reaction:** The action is additionally recorded in the list of actions. ● Should it be necessary to overwrite the current action with the new action, use the command button **Accept** **Reaction:** The old action is overwritten.

Deleting an action declaration

The procedure for deleting an action declaration is as follows:

Step	Action
1	With **Objects** →**Selection mode** go to selection mode.
2	Double-click on a step. **Reaction:** The dialog **Step properties** of the step is opened.
3	To delete an action declaration, select an action in the list. **Reaction:** All definitions (identifiers, time duration, variable or address and type) of the action are transferred into the corresponding text boxes and lists.
4	Use the command button **Delete**. **Reaction:** The selected action is deleted.

Identifier

At a Glance

For every connection of an action to a step, an identifier must be defined for the action. The identifier must define the control of the action. The identifier can be introduced as the input of an internal Function Block for the configured link of the step with the action. If the step is active, the input of this internal Function Block is set to 1. The Function Block is then processed according to its type. If all conditions are satisfied, the output Q (action) is set to 1.

The following identifiers are usable in Concept:
- N / none *(see page 302)*
- S *(see page 302)*
- R *(see page 303)*
- L *(see page 303)*
- D *(see page 303)*
- P *(see page 304)*
- DS *(see page 304)*

For the identifiers L, D and DS, a time duration of the data type TIME must additionally be defined.

Identifier N / none

The identifiers N and none have the same meaning and stand for "Not saved" and/or "No identifier".

Identifier S

The identifier S stands for "set (saved)".

The set action also remains active, when the associated step is inactive. The action first becomes inactive, when reset is used with the Identifier R *(see page 303)* in another step.

NOTE: The identifier is automatically declared as unbuffered. This means that the value is reset to "0" after stop and cold restart, e.g. when voltage is on/off. Should a buffered output be required, please use the RS or SR Function Block from the IEC block library.

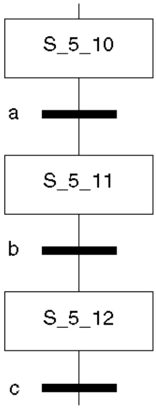
Identifier R

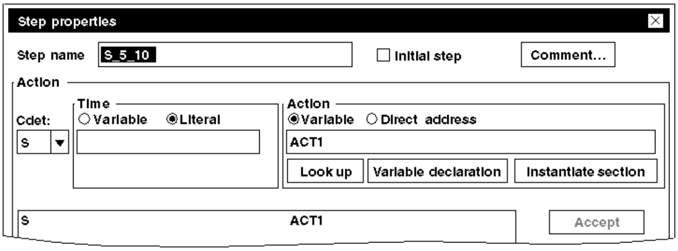
The identifier R stands for "overriding reset".

The action, which is set in another step with the Identifier S *(see page 302)*, is reset. The activation of any action can also be prevented.

NOTE: The identifier is automatically declared as unbuffered. This means that the value is reset to "0" after stop and cold restart, e.g. when voltage is on/off. Should a buffered output be required, please use the RS or SR Function Block from the IEC block library.

In the step S_5_10 the action ACT1 becomes and remains active, until the reset in step S_5_12.




Step properties

Step name: S_5_10 □ Initial step Comment...

Action

Cdet: S

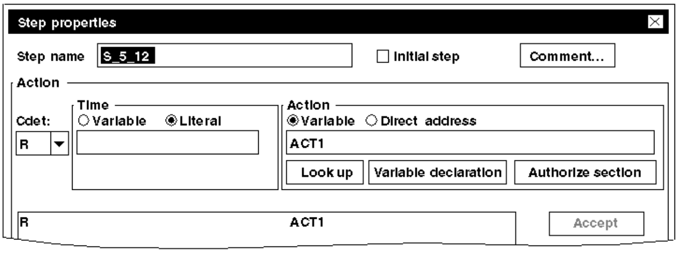
Time: ○ Variable ● Literal

Action: ● Variable ○ Direct address

ACT1

Look up | Variable declaration | Instantiate section

S ACT1 Accept


Step properties

Step name: S_5_12 □ Initial step Comment...

Action

Cdet: R

Time: ○ Variable ● Literal

Action: ● Variable ○ Direct address

ACT1

Look up | Variable declaration | Authorize section

R ACT1 Accept

Identifier L

The identifier L stands for "Limited".

If the step is active, the action is also active. After the process of the time duration, defined manually for the action, the action returns to 0, even if the step is still active. The action also becomes 0 if the step is inactive.

Identifier D

The identifier D stands for "delayed".

If the step is active, the internal timer is started and the action becomes 1 after the process of the time duration, which was defined manually for the action. If the step becomes inactive after that, the action becomes inactive as well. If the step becomes inactive before the process of the internal time, the action does not become active.

Identifier P

The identifier P stands for "Pulse".

If the step becomes active, the action becomes 1 and this remains for one program cycle, independent of whether or not the step remains active.

Identifier DS

The identifier DS stands for "delayed and saved". It is a combination of the identifiers D *(see page 303)* and S *(see page 302)*.

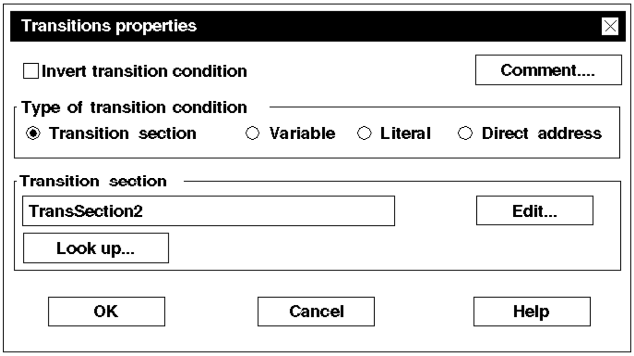
If the step becomes active, the internal timer is started and the action becomes active after the process of the manually defined time duration. The action first becomes inactive once again, when reset is used with the IdentifierR *(see page 303)* in another step. If the step becomes inactive before the process of the internal time, the action does not become active.

Declaring a Transition

Introduction

Transitions are declared in the properties dialog of the transition.

Declaring a transition:



Transitions properties

□ Invert transition condition

Type of transition condition

● Transition section ○ Variable ○ Literal ○ Direct address

Transition section

TransSection2

Declaring a transition:

The following example describes the procedure when declaring a transition:

Step	Action
1	With **Objects** → **Selection mode** go to selection mode.
2	Double-click on a transition. **Response:** The dialog **Transition properties** of the transition is opened.
3	Begin by determining **Kind of transition condition:** determine the type (**Transition section**, **Variable**, **Literal**, **Direct address**) of transition condition.
4	● After selecting the**Transition section** has been selected, enter in the text box **Transition section** the name of the transition section to be created. This is a section containing the logic of the transition condition and it is automatically linked with the transition. To process this section, press the command button **Process...**. ● After selecting the**Variable** has been selected, enter in the text box **BOOL variable** the name of the selected unlocated variable, located variable or constants. **Note:** For an example for invocation of multi-element variables see *Calling Derived Data Types, page 588*. ● If the **Literal** has been selected, select in the field **Value** the value of the literal. ● If the **Dir. address** , enter in the text box **Direct address**the required address.
5	The transition condition can now be inverted with the **Invert trans. cond.** check box. **Response:** An inverted transition condition is displayed with a (~) symbol in front of the name of the variable on the transition.
6	With the command button **Comment** click on the dialog box **Enter with comment**, in which a comment about the transistion can be entered. This comment is shown in the status bar of the editor window, if the transition is selected.
7	After all the definitions for the transition have been met, confirm this with the command button **OK**.

Copying transition conditions

By copying and inserting it is possible to copy transitions through projects. Since the definition of a transition displays a reference to a variable, which is defined by the Variable Editor for the particular project, copying between projects can result in this reference no longer being valid. In this instance, the transition condition is deleted and an error message appears.

Alias Designations for Steps and Transitions

Introduction

Instead of free names you can also select alias designations for steps and transitions. These are then displayed in SFC and FBD sections during search functions, application documentation and analysis.

Import and export functions do not recognize the alias designations, since they are dynamically generated. The visualization can retrieve the alias designations dynamically, however they cannot be used for the configuration of fixed references, since they can change constantly.

The languages ST, IL and LD do not support alias designations and display the free names.

Name Definition

The alias designations are dynamically generated during editing procedures, and the same applies when the **Dynamic Numbered** option is switched on.

Alias designations remain empty until numbering can take place i.e. when all objects are linked to one chain.

The alias designations are made up of the position of the steps and transitions in the section and the section name.

The length of the section name part displayed in the alias designation is freely definable in the **Options** →**Preferences** →**Graphical Editors Preferences** dialog. You can define how many characters from the section name (beginning with the first character) should go into the alias designations here.

NOTE: The settings in this dialog are used in the project description (PRJ.DSK) and in the Concept installations description (CONCEPT.DSK), i.e. they are valid for the entire Concept installation.

If a project is opened, which was created using alternative settings (e.g. Settings from **Presentation of Steps and Transitions** numbered **IEC_like** in the project and **Dynamic numbered** in the current Concept installation), errors can occur when opening projects.

Alias Designations for Steps

With steps, the lines and columns occupied by steps are each numbered beginning with the top left. A four-figure step number is made from the column and line numbers (ccll). The alias designation for steps is made from S_ string, part of the section name (nnn), a further underscore (_) and the step number (ccll) (S_nn_ccll).

Alias Designations for Transitions

The alias designations for transitions are derived from the alias designation of the preceding step cell, even when this is empty. The alias designation for transitions is made from the T_ string, part of the section name (nnn), a further underscore (_) and the number of the preceding step cell (ccll) (T_nn_ccll).

Activating the Alias Designations

The free name is entered as the default for steps and transitions. If you require alias designations, you can activate them in the **Options** →**Preferences** →**Graphical Editors Preferences** dialog using the **Dynamic Numbered** option.

⚠ CAUTION

Danger of loss of data.

The free names (**IEC_like**) are overwritten by the alias names when this option is selected. If you want to restore the free names, close the project without saving.

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

⚠ CAUTION

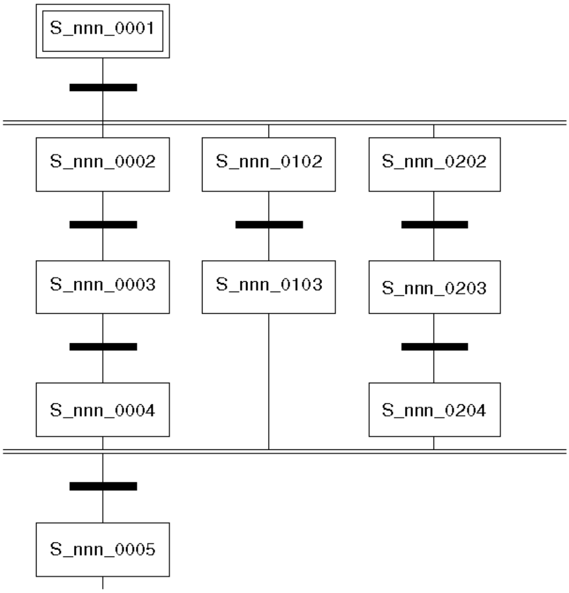
Danger of loss of data.

You must not switch between the **IEC_like** and **Dynamic Numbered** display modes if an FBD transition section is already open. Otherwise, this could result in section and variable names containing spaces. Therefore, close all FBD transition sections before you change the representation mode.

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

Example for Alias Designations

Example for alias designations:



Inserting and Deleting Objects

When inserting and deleting objects (steps and transitions) the alias designations are renumbered.

9.4 Online functions of the SFC sequence language

Overview

This section describes the online functions of the SFC sequence language elements.

What's in this Section?

This section contains the following topics:

Topic	Page
Animation	311
Controlling a Step String	313
Learn monitoring times	316
Transition diagnosis	319

Animation

Introduction

In the animation mode the following are displayed in different colors in the editor window:

- the active steps,
- the time the steps are or were active for,
- time out errors of the steps and
- the status of the transitions (made, not made).

NOTE: If the transition, and therefore the transition section, is not processed, the status **DISABLED** appears in the animated transition section.

Activating the Animation

The animation is activated with the menu command **Online →Animation**.

Color key

There are 12 different color schemes available for animation. An overview of the color scheme and the meaning of each color can be found in Online help (Tip: Search the online help for the index reference "Colors").

Changing Values

In this mode the following can be changed:

- With transitions:
  - the transition condition, if this is a literal.
- With steps:
  - the maximum supervision time,
  - the minimum supervision time,
  - the delay time and
  - the times of the actions.

These changes are sent online to the PLC.

Transition Animation

Normally, only the currently evaluated transitions are animated and their status (transition condition satisfied/not satisfied) is displayed.

It is also possible to display the status of the transitions not currently being processed. This will only show the status of the transitions. It has no influence on the behavior of the sequence. To do this you require the XSFCCNTRL function block of the SYSTEM block library. Additionally, in the **Options** →**Preferences** →**Graphical Editors** dialog, you must check the **Animate All Conditions of the Transition Section** check box.

NOTE: This function leads to a considerable burden on the logic scan. This results from the fact that all the transitions in the affected section are solved and animated in one logic scan, whereas this is normally solved sequentially depending on the process status (preceding step active/inactive).

Displaying all Transition Conditions

The procedure for showing all transition conditions is as follows:

Step	Action
1	Create an FBD section and enter the XSFCCNTRL function block of the SYSTEM block library.
2	Enter the names of the SFC section to be animated as the instance name (block name) of the XSFCCNTRL function block.
3	Assign the value "1" to the ALLTRANS input of the XSFCCNTRL function block (using a literal or, depending on the process, a variable). **Response:** By doing this, the calculation of all transition conditions is activated. Otherwise an old status of the transition condition would be displayed.
4	With the menu command **Project** →**Execution Order...** (or the project browser) ensure that the FBD section is executed before the SFC section to be animated
5	Check the **Animate All Conditions of the Transition Section** check box in the **Options** →**Preferences** →**Graphical Editors** dialog.
6	Download the program to the PLC and start the animation of the SFC section. **Response:** All transition conditions are then displayed.

Controlling a Step String

Introduction

There are 3 ways of controlling a string:

- with the animation control
- with the menu commands in the main menu **Online**
- with the SFCCNTRL or XSFCCNTRL function block (SYSTEM block library)

If controlling a string through the different options simultaneously, these control operations have equal priority.

The control operations triggered using the menu commands in the **Online** main menu and using the animation panel can be locked by the function blocks SFCCNTRL and XSFCCNTRL.

A control operation in one of the methods is also displayed in the other two methods.

Requirements

It is only possible to control the step string when the animation mode for the section is active.

Animation Panel

The animation panel is activated with the menu command **Online** →**Show Animation Panel**.

The animation panel contains all the possibilities that are also available as menu commands.

Mode of Functioning

You can test the processing of an SFC section with the animation panel and the menu commands. For example, steps can be relayed, the processing of the string can be controlled (whether or not transitions and/or actions are to be processed), time errors can be reset or the string can be reset to initialization status.

⚠ WARNING

Danger of unsafe, dangerous and destructive tool operations.

Set/Reset flag, **Disable Transitions**, **Disable Actions**, **Step Unconditional**, **Step/Trans. dependant** and **Force Selected Steps** should not be used for debugging on controllers of machine tools, processes or material maintenance systems when they are running. This can lead to unsafe, dangerous and destructive operation of tools or processes linked to the controller.

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

Set/Reset Flag

The **Set/Reset flag** resets the string and starts it as standard.

- **Reset chain**
  To reset the string, activate **Set/Reset Flag**. This stops the chain and all actions are reset. No operator interventions are possible.
- **Starting the chain in a standardized way**
  For a standardized start of the string, **Set/Reset Flag** must first be activated and then deactivated. With the 1 →0 slope the chain is reset i.e. the initial step is activated.

Disable Time Check

If **Disable Time Check** is activated, there is no longer any time supervision of the steps. The step delay time, however, still remains active.

Disable Transitions

If **Disable Transitions** is activated, the transition conditions are no longer utilized. The string remains in its current state, independent of the signals on the transitions. The string can still only be used via the control commands (**Set/Reset Flag**, **Step Unconditional**, **Step/Trans. Dependant**).

Disable Actions

If **Disable Actions** is activated, the step actions are no longer processed.

Step Unconditional

The next step is activated independently of the transition status, but not until the step delay time of the active step has elapsed.

With **Step Unconditional**, all branches are activated in parallel branches, and the left branch is always activated in alternate branches.

Step/Trans. dependent is used for activating process-dependent branches.

⚠ WARNING

Danger of unsafe, dangerous and destructive tool operations.

Step Unconditional activates the next step, even if the transition is not satisfied.

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

Step/Trans. Dependent

The next step is activated when the transition conditions are satisfied.

Step/Trans. Dependent is advisable only when **Disable Transitions** is active.

By freezing the transitions (**Disable Transitions**) it is possible, with **Step/Trans. Dependent** to process the string elements manually step by step. In this way the transitions commutate depending on the transition condition.

Reset Time Error

If **Reset Time Error** is activated, the error message display for time supervision in the SFC section is reset.

Force Selected Steps

The selected step(s) are activated independent of the status of the transitions and steps.

In alternative branches, only one single step and one single branch can be activated.

In parallel branches, steps can only be set, if the process is already located in the parallel branch and one step in every branch is active. If one step is set in a parallel branch, all other parallel branches remain unaffected by it.

⚠ WARNING

Danger of unsafe, dangerous and destructive tool operations.

Force Selected Steps activates the selected steps, even if the transition is not satisfied.

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

This functionality is not available via the function blocks SFCCNTRL or XSFCCNTRL (SYSTEM block library).

Select Active Steps

The active step of the step string is searched for and selected.

Learn monitoring times

At a Glance

In this mode, the minimum and maximum times, for which the steps were active, are determined. After mode deactivation, the determined times for the single steps are shown in the **Learn step monitoring times** dialog box. From there, the minimum *(see page 276)* and maximum monitoring time *(see page 275)* are accepted in the step properties. During the transfer, a factor can be specified for the minimum and maximum time.

NOTE: This functionality is not available via the Function Blocks SFCCNTRL or XSFCCNTRL (Block library SYSTEM).

Note on determining values

Please ensure that at least 2 cycles typical for the process were gathered.

The determined values are first saved after the single step becomes inactive, i.e. if a step was never active during the "Learn monitoring times" mode, no value is determined for this step.

The storage of all determined step cells of a cycle can take some time. Because of this, very long step sequence times and very short individual step durations may be indeterminable, due to internal time overlaps.

Use of ’SFCSTEP_TIMES’ variable or constants

Should the step have been assigned a ’SFCSTEP_TIMES’ variable or constant in the **Step properties** dialog, the times learned for these variables/constants are shown as the initial value. Should these initial values be used for a long period of time, do not allow corresponding elements (min., max.) of these variables/constants to be written.

After learning the monitoring times, the altered initial values must be loaded into the PLC.

- This is performed for variables with the menu command **Online →Load**.
- This is performed for constants with the menu command **Online →Load changes.**

Calculating "learned" times

A factor can be defined for the determined values, which are multiplied when calculating the monitoring times.

- Minimum monitoring time = minimum determined time x Minimum [%]
- Maximum monitoring time = maximum determined time x Maximum [%]

Calculating "learned" times " Example 1

Calculating "learned" times

- The determined times for one step are: 1 s, 2 s, 2 s
- Minimum [%]: 50
- Maximum [%]: 200

Following the above formula, this results in a minimum monitoring time of 500 ms and a maximum monitoring time of 4 s.

Calculating "learned" times " Example 2

If a delay time is given for the step, this is considered when calculating the minimum monitoring time. I.e. if the delay time is larger than the calculated value for the minimum monitoring time, the calculated value for the minimum monitoring time is ignored and set to 0 ms (i.e. there is no monitoring of the minimum time).

Calculating "learned" times

- The determined times for one step are: 1 s, 2 s, 2 s
- Delay time: 2 s
- Minimum [%]: 50
- Maximum [%]: 200

This results in a minimum monitoring time of 0 ms and a maximum monitoring time of 4 s.

Calculating "learned" times " Example 3

If a delay time is given for the step, this is likewise considered when calculating the maximum monitoring time. I.e. if the delay time is larger than the calculated value for the maximum monitoring time, the calculated value for the maximum monitoring time is ignored and in its place, a suitable value is calculated.

In such a case 2 cases are considered:

- A value for the minimum monitoring time is available.
  Then the value for the maximum monitoring time is calculated according to the following formula: Minimum monitoring time + 20 ms
  Example:
  - The determined times for one step are: 2 s, 2 s, 2 s
  - Delay time: 3 s
  - Minimum [%]: 200
  - Maximum [%]: 100

  Following the above formula, this results in a minimum monitoring time of 4 s and a maximum monitoring time of 4s20ms.

- No value for the minimum monitoring time is available, see *example 2*.
  Then the value for the maximum monitoring time is calculated according to the following formula: Delay time + 20 ms
  Example:
  - The determined times for one step are: 1 s, 2 s, 2 s
  - Delay time: 1 s
  - Minimum [%]: 50
  - Maximum [%]: 100

  Following the above formula, this results in a minimum monitoring time of 0 s and a maximum monitoring time of 1s20ms.

Transition diagnosis

Preview

The transition diagnosis monitors that the immediately preceding step was active following the transition, commutated within a certain time in the step sequence (with parallel branches in the step sequences). Should this not be the case, the associated transition network (with alternative branches, the transition network of all associated transitions) is analysed, and the error, including the analysed signal, is entered in the signal buffer. This can now be evaluated using visualization software (e.g. MonitorPro, Factory Link).

NOTE: The transition diagnosis only runs when the string is active.

Transition diagnosis vs. Reaction diagnosis

The performance of the transition diagnosis is about equal to that of the reaction diagnosis (see *Function Block REA_DIA from the block library DIAGNO*). Contrary to the reaction diagnosis the re-registration of all the actions started and possible additional conditions are monitored here.

Activating the transition diagnosis

Activating the transition diagnosis:

Step	Action
1	Activate the transition diagnosis by entering a Mon. time in the field **Maximum** step properties of the immediately preceding step (see also *Learn monitoring times, page 316*). If the field remains empty or the time 0 is entered the transition monitoring is inactive.
2	Aktivate in the dialog **Project →Code generation options... →Code generation options...** the option **Include diagnosis information** to make memory available in the PLC for the error buffer.
3	Load the altered configuration into the PLC.

Instruction list IL

10

Overview

This Chapter describes the programming language instruction list IL which conforms to IEC 1131.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
10.1	General information about the IL instruction list	322
10.2	Instructions	324
10.3	IL instruction list operators	339
10.4	Call up of functions, Function Blocks (EFBs) and Derived Function Blocks (DFBs)	371
10.5	Syntax check and Code generation	381
10.6	Online functions of the IL instruction list	385
10.7	Creating a program with the IL instruction list	390

10.1 General information about the IL instruction list

General Information about the IL Instruction List

Introduction

With assistance from the programming language (IL) instruction list e.g. Function Blocks and functions can be called up conditionally or unconditionally, assignments can be performed, and jumps can be performed conditionally or unconditionally within a section.

Spell Check

Spelling is immediately checked when key words, separators and comments are entered. If a key word, separator or comment is recognized, it is identified with a color surround. If unauthorized key words (instructions or operators) are entered, it is likewise identified in color.

IEC Conventions

The IEC 1131 does not permit the input of direct addresses in the usual Concept form. To input direct addresses see *Operands, page 327*.

In accordance with IEC 113-3, key words must be entered in upper case. Should the use of lower case letters be required, they can be enabled in the dialog box **Options →Preferences →IEC Extensions... →IEC expansions** with the option **Allow case insensitive keywords**.

Blank spaces and tabs have no influence upon the syntax and can be used freely.

Context help

With the right mouse button an object can be selected and at the same time a context sensitive menu called up. Therefore, for example, with FFBs the right mouse button can call up the associated block description.

Syntax Check

A syntax check can be performed during the program/DFB creation with **Project → Analyze section**, see also *Syntax Check, page 382*.

Codegeneration

Using the **Project →Code Generation Options** menu command, you can define options for code generation, see also *Code generation, page 384*.

Editing with the Keyboard

Normally editing in Concept is performed with the mouse, however it is also possible with the keyboard (see also *Short Cut Keys in the IL, ST and Data Type Editor, page 834*).

IEC Conformity

For a description of the IEC conformity of the IL programming language see *IEC conformity, page 849*.

10.2 Instructions

Overview

This section contains an overview of the instructions for the programming language instruction list.(IL)

What's in this Section?

This section contains the following topics:

Topic	Page
General information about instructions	325
Operands	327
Modifier	329
Operators	331
Tag	334
Declaration (VAR...END_VAR)	336
Comment	338

General information about instructions

At a Glance

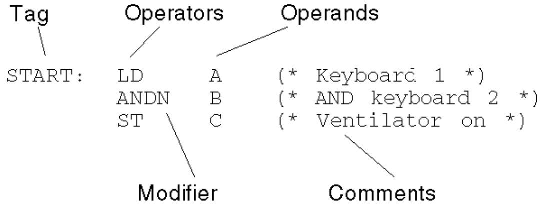
An instruction list is composed of a series of instructions.

Each instruction begins on a new line and consists of:

- an Operator *(see page 327)*,
- if necessary with modifier *(see page 329)* and
- if necessary one or more operands *(see page 339)*.

Should several operands be used, they are separated by commas. It is possible for a mark *(see page 334)*to be in front of the instruction, which is followed by a colon. A comment *(see page 338)* can follow the instruction.

Example:



```
START:  LD     A    (* Keyboard 1 *)
        ANDN   B    (* AND keyboard 2 *)
        ST     C    (* Ventilator on *)
```

Structure of the programming language

IL is a so-called battery orientated language, i.e. each instruction uses or alters the current content of the battery (a form of internal cache). The IEC 1131 refers to this battery as the "result".

For this reason, an instruction list should always begin with the LD operand ("Load in battery command").

Example of an addition:

Command	Meaning
`LD 10`	The value "10" is loaded into the battery.
`ADD 25`	"25" is added to the battery content.
`ST A`	The result is stored in the "A" variable. The content of the "A" variable and the battery is now "35". A possible ensuing instruction would be worked with the battery content "35", should it not begin with LD.

Comparative operations likewise always refer to the battery. The Boolean result of the comparison is stored in the battery and is therefore the current battery content.

Example of a comparison:

Command	Meaning
`LD B`	The value "B" is loaded into the battery.
`GT 10`	"10" is compared with the battery content.
`ST A`	The result of the comparison is stored in the "A" variable. If "B" is less than or equal to "10", the value of both the "A" variable, and the battery content is "0" (FALSE). If "B" is greater than or equal to "10", the value of both the "A" variable, and the battery content is "1" (TRUE).

Operands

At a Glance

An operand can be:

- a literal,
- a variable,
- a multi-element variable,
- an element of a multi-element variable,
- a FB/DFB output or
- a direct address.

Access to the field variables

When accessing the field variable (ARRAY), only literals and variables of ANY_INT type are permitted in the index entry.

Example: Saving a field variable

```
LD var1[i]
ST var2.otto[4]
```

Type conversion

The operand and the current accu content must be of the same type. Should operands of various types be processed, a type conversion must be performed beforehand.

An exception is the data type TIME in conjunction with the arithmetic operators MUL and DIV. With both these operators, an operand of TIME data type can be processed together with an operand of ANY_NUM data type. The result of this instruction has in this instance the data type TIME.

Example: Integer variable and real variable

In the example the integer variable "i1" is converted into a real variable, before being added to the real variable "r4".

```
LD i1
INT_TO_REAL
ADD r4
ST r3
```

Example: Integer variable and time variable

In the example the time variable "t2" is multiplied by the integer variable "i4" and the result is stored in the time variable "t1".

```
LD t2
MUL i4
ST t1
```

Default data types of direct addresses

The following table shows the default data types of direct addresses:

Input	Output	Default data type	possible data type
%IX,%I	%QX,%Q	BOOL	BOOL
%IB	%QB	BYTE	BYTE
%IW	%QW	INT	INT, UINT, WORD
%ID	%QD	REAL	REAL, DINT, UDINT, TIME

Using other data types

Should other data types be assigned as default data types of a direct address, this must be done through an explicit declaration (VAR…END_VAR *(see page 336)*). VAR…END_VAR cannot be used in Concept for the declaration of variables. The variable declaration conveniently follows the Variable Editor *(see page 543)*.

Modifier

At a Glance

Modifiers influence the implementation of the preceding operators (see *Operators, page 331*).

Modifier N

The Modifier N is used to invert the value of the operands bit by bit.

The modifier can only be used on operands with the ANY_BIT data type.

Example: N

In the example C will be "1", when A is "1" and B is "0".

```
LD A
ANDN B
ST C
```

Modifier C

The modifier C is used to carry out the associated instruction, should the value of the battery be "1" (TRUE).

The modifier can only be used on operands with the BOOL data type.

Example: C

In the example the jump after START is only performed, when A is "1" (TRUE) and B is "1" (TRUE).

```
LD AAND BJMPC START
```

Modifier CN

If the modifiers C and N are combined, the associated instruction is only performed, should the value of the battery be a Boolean "0" (FALSE).

Example: CN

In the example, the jump after START is only performed, when A is "0" (FALSE) and/or B is "0" (FALSE).

```
LD A
AND B
JMPCN START
```

Left bracket modifier "("

The left bracket modifier "(" is used to move back the evaluation of the operand, until the right bracket operator appears. The number of right bracket operations must be equal to the number of left bracket modifiers. Brackets can be nested.

Example: Left bracket "("

In the example E will be "1", if C and/or D is "1", just as A and B are "1".

```
LD A
AND B
AND( C
OR D
)
ST E
```

The example can also be programmed in the following manner:

```
LD A
AND B
AND(
LD C
OR D
)
ST E
```

Operators

At a Glance

An operator is a symbol for:

- an arithmetic operation to be executed,
- a configured operation to be executed or
- the function call up.

Operators are generic, i.e. they are automatically suited to the operands data type.

NOTE: Operators can be either entered by hand or generated with assistance from the menu **Objects**.

Table of operators

IL programming language operators:

Operator	**Operator key**	**possible modifier**	**possible operand**	**see also**
LD	Loads the operands value into the battery	N	Literal, variable, direct address of ANY data type	*Load (LD and LDN), page 340*
ST	Saves the battery value in the operand	N	Variable, direct address of ANY data type	*Store (ST and STN), page 341*
S	Sets the operand to 1, when the battery content is 1		Variable, direct address of BOOL data type	*Set (S), page 342*
R	Sets the operand to 0, when the battery content is 1		Variable, direct address of BOOL data type	*Reset (R), page 344*
AND	Configured AND	N, N(, (	Literal, variable, direct address of ANY_BIT data type	*Boolean AND (AND, AND (), ANDN, ANDN ()), page 346*
OR	Configured OR	N, N(, (	Literal, variable, direct address of ANY_BIT data type	*Boolean OR (OR, OR (), ORN, ORN ()), page 348*
XOR	Configured exclusive OR	N, N(, (	Literal, variable, direct address of ANY_BIT data type	*Boolean exclusive OR (XOR, XOR (), XORN, XORN ()), page 350*

Operator	Operator key	possible modifier	possible operand	see also
ADD	Addition	(	Literal, variable, direct address of ANY_NUM data type or TIME data type	*Addition (ADD and ADD ()), page 353*
SUB	Subtraction	(	Literal, variable, direct address of ANY_NUM data type or TIME data type	*Subtraction (SUB and SUB ()), page 354*
MUL	Multiplication	(	Literal, variable, direct address of ANY_NUM data type or TIME data type	*Multiplication (*), page 408*
DIV	Division	(	Literal, variable, direct address of ANY_NUM data type or TIME data type	*Division (DIV and DIV ()), page 357*
GT	Comparison: >	(	Literal, variable, direct address of ANY_ELEM data type	*Compare on "Greater Than" (GT and GT ()), page 359*
GE	Comparison: >=	(	Literal, variable, direct address of ANY_ELEM data type	*Compare to "Greater than/Equal to" (GE and GE ()), page 360*
EQ	Comparison: =	(	Literal, variable, direct address of ANY_ELEM data type	*Compare to "EQual to"(EQ and EQ ()), page 361*
NE	Comparison: <>	(	Literal, variable, direct address of ANY_ELEM data type	*Compare to "Not Equal to" (NE and NE ()), page 362*
LE	Comparison: <=	(	Literal, variable, direct address of ANY_ELEM data type	*Compare to "Less than/Equal to" (LE and LE ()), page 363*
LT	Comparison: <	(	Literal, variable, direct address of ANY_ELEM data type	*Compare to "Less Than"(LT and LT ()), page 364*

Operator	**Operator key**	**possible modifier**	**possible operand**	**see also**
JMP	Jump to tag	C, CN	TAG	*Jump to label (JMP, JMPC and JMPCN), page 365*
CAL	Calling up a Function Block or DFB	C, CN	FBNAME (item name)	*Call Function Block/DFB (CAL, CALC and CALCN), page 368*
FUNCNAME	Performing a function		Literal, variable, direct address (data type is dependent on function)	*Function call, page 379*
)	Editing on-hold operations			*Right parenthesis ")", page 370*

Tag

At a Glance

Tags serve as destinations for Jumps *(see page 365)*.

Properties

Tag properties:

- Tags must always be the first element in a line.
- Tags must be unique throughout the project/DFB, and are not case-sensitive.
- Tags can be 32 characters long (max.).
- Tags must conform to the IEC name conventions.
- Tags are separated by a colon ":" from the following instruction.
- Tags are only permitted at the beginning of "Expressions", otherwise an undefined value can be found in the battery.

Destinations

Possible destinations are:

- the first LD instruction of a FB/DFB call up with assignment of input parameters (see `start2`),
- a normal LD instruction (see `start1`),
- a CAL instruction, which does not work with assignment of input parameters (see`start3`),
- a JMP instruction (see `start4`),
- the end of an instruction list (see `start5`).

Example

```
start2: LD A
        ST counter.CU
        LD B
        ST counter.R
        LD C
        ST counter.PV
        CAL counter
        JMPCN start4
start1: LD A
        AND B
        OR C
        ST D
        JMPC start3
        LD A
        ADD E
        JMP start5
start3: CAL counter (
        CU:=A
        R:=B
        PV:=C )
        JMP start1
start4: JMPC start1
start5:
```

Declaration (VAR...END_VAR)

At a Glance

The VAR instruction is used to declare the function blocks and DFBs used, and direct addresses if they are not to be used with the default data type. VAR cannot be used for declaring a variable in Concept. Declaring the variables may conveniently be done via the Variables editor.

The END_VAR instruction marks the end of the declaration.

NOTE: The declaration of the FBs/DFBs and direct addresses applies only to the current section. If the same FFB type or the same address are also used in another section, the FFB type or the address must be declared again in this section.

Declaration of function blocks and DFBs

During declaration for each FB/DFB example, a unique example name is assigned. The example name is used to mark the function block uniquely in a project. The example name must be unique in the whole project; no distinction is made between upper/lower case. The example name must correspond to the IEC Name conventions, otherwise an error message will be displayed.

After specifying the example name, the function block type, e.g.CTD_DINT is specified.

In the case of function block types no data type is specified. It is determined by the data type of the actual parameters. If all actual parameters consist of literals, a suitable data type will be selected.

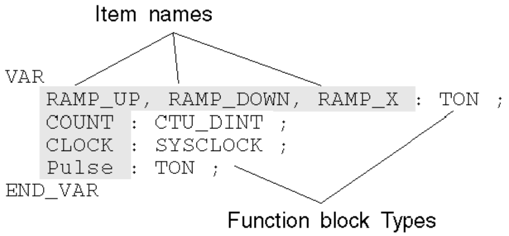
Any number of example names may be declared for an FB/DFB.

NOTE: The dialog **Objects →Insert FFB** provides you with a form for creating the FB-/DFB declaration in a simple and speedy manner.

NOTE: In contrast to grafic programming languages (FBD, LD), it is possible to call up multiple calls in FB/DFB examples within IL.

Example

Declaration of function blocks and DFBs



```
VAR
   RAMP_UP, RAMP_DOWN, RAMP_X : TON ;
   COUNT : CTU_DINT ;
   CLOCK : SYSCLOCK ;
   Pulse : TON ;
END_VAR
```

Declaration of direct addresses

In the case of this declaration, every direct address used whose data type does not correspond to the default data type will be assigned the required data type (see also Default data types of direct addresses *(see page 328)*).

Example

Declaration of direct addresses

```
VAR
   AT %QW1 : WORD ;
   AT %IW15 : UINT ;
   AT %ID45 : DINT ;
   AT %QD4 : TIME ;
END_VAR
```

Comment

Description

Within the IL Editor, comments always start with the string (* and end in the string *). Any comments may be entered between these two strings. Comments are shown in colors.

NOTE: In accordance with IEC 1131-1, comments are only permissible at the end of a line. However, if you wish to place theses elsewhere, you can do this by using **Options →Preferences →IEC Extensions →Allow comments anywhere in text (IL)**.

NOTE: In accordance with IEC 1131-1, nested comments are not permissible. However, if you wish to place theses elsewhere, you can do this by using **Options →Preferences →IEC Extensions →Allow nested comments**.

10.3 IL instruction list operators

Overview

This section describes the IL instruction list operators.

What's in this Section?

This section contains the following topics:

Topic	Page
Load (LD and LDN)	340
Store (ST and STN)	341
Set (S)	342
Reset (R)	344
Boolean AND (AND, AND (), ANDN, ANDN ())	346
Boolean OR (OR, OR (), ORN, ORN ())	348
Boolean exclusive OR (XOR, XOR (), XORN, XORN ())	350
Invert (NOT)	352
Addition (ADD and ADD ())	353
Subtraction (SUB and SUB ())	354
Multiplication (MUL and MUL())	355
Division (DIV and DIV ())	357
Compare on "Greater Than" (GT and GT ())	359
Compare to "Greater than/Equal to" (GE and GE ())	360
Compare to "EQual to"(EQ and EQ ())	361
Compare to "Not Equal to" (NE and NE ())	362
Compare to "Less than/Equal to" (LE and LE ())	363
Compare to "Less Than"(LT and LT ())	364
Jump to label (JMP, JMPC and JMPCN)	365
Call Function Block/DFB (CAL, CALC and CALCN)	368
FUNCNAME	369
Right parenthesis ")"	370

Load (LD and LDN)

LD Description

With LD the value of the Operands is downloaded into the accumulator. The data width of the accumulator adapts itself automatically to the data type of the operand. This also applies to derived datatypes.

Example LD

Example LD

Operation	Description
LD A	The value of "A" is downloaded onto the accu.
ADD B	The accu contents are added to the value of "B".
ST E	The result is saved in "E".

LDN Description

The downloaded operand can be negated with the Modifier N (only if the Operand is of data type ANY_BIT).

LDN Example

LDN Example

Operation	Description
LDN A	The value of "A" is inverted and downloaded onto the accu.
ADD B	The accu contents are added to the value of "B".
ST E	The result is saved in "E".

Store (ST and STN)

ST Description

With ST the current value of the accu is saved in the operand. The data type of the operand must therefore agree with the "data type" of the accu.

Depending on whether an LD follows after ST or not, calculation proceeds with the "old" result.

ST Example

ST Example

Operation	Description
`LD A`	The value of "A" is downloaded onto the accu.
`ADD B`	The accu contents are added to the value of "B".
`ST` `E`	The result is saved in "E".
`ADD B`	Afterwards the value of "E" (current accu contents) is added to the value of "B" again
`ST` `F`	The result is saved in "F".
`LD` `X`	The value of "X" is now downloaded onto the accu.
`SUB 3`	3 is subtracted from the accu contents.
`ST` `Y`	The result is saved in "Y".

STN Description

The operand to be saved can be negated with the N modifier (only if the operand is on the ANY_BIT data type).

STN Example

ST Example

Operation	Description
`LD A`	The value of "A" is downloaded onto the accu.
`ADD B`	The accu contents are added to the value of "B".
`STN` `E`	The result is inverted and saved in "E".

Set (S)

Description

S sets the operand to "1" when the current content of the accu is a Boolean "1".

Example S

Example S

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`S OUT`	If the content of the accu (the value of A) is "1", "OUT" is set to "1".

Use

Usually this operator is used together with the Reset operator R (flip flop) as a pair.

Example RS flip flop

The example shows an RS flip flop (Reset dominant).

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`S OUT`	If the content of the accu (the value of "A") is "1", "OUT" is set to "1".
`LD C`	The value of "C" is loaded into the accu.
`R OUT`	If the content of the accu (the value of "C") is "1", "OUT" is set to "0".

Start behavior

The start behavior of PLC’s is divided into cold and warm starts.

- **Cold Start**
  Following a cold start (loading the program with **Online** →**Download**) all variables are set (independently of their type) to "0" or, if available, to their initial value.
- **Warm Start**
  On a warm start (stopping and starting of the program or **Online** →**Download changes**) different start behavior applies for located variables/direct addresses and unlocated variables:
  - **Located variables/direct addresses**
    During a warm start the located variable/direct address, is set to "0", or to its initial value if present, via the set instruction.
  - **Unlocated variables**
    On a warm start the unlocated variable, set via the set instruction, maintains its present value (storing behavior).

NOTE: Should a buffered located variable/direct address be required, please use the RS or SR function blocks from the block library IEC.

Reset (R)

Description

R sets the operand to "0" when the current content of the accu is a Boolean "1".

Example R

Example R

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`R` `OUT`	If the content of the accu (the value of "A") is "1", "OUT" is set to "0".

Use

Usually this operator is used together with the Set operator S (flip flop) as a pair.

Example SR flip flop

The example shows an SR flip flop (Set dominant).

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`R` `OUT`	If the content of the accu (the value of "A") is "1", "OUT" is set to "0".
`LD C`	The value of "C" is loaded into the accu.
`S OUT`	If the content of the accu (the value of "C") is "1", "OUT" is set to "1".

Start behavior

PLC start behavior is divided into cold and warm starts:

- **Cold Start**
  Following a cold start (loading the program with **Online** →**Download**) all variables are set (independently of their type) to "0" or, if available, to their initial value.
- **Warm Start**
  On a warm start (stopping and starting of the program or **Online** →**Download changes**) different start behavior applies for located variables/direct addresses and unlocated variables:
  - **Located variables/direct addresses**
    On a warm start the located variable/direct address, is set to "0", or to its initial value if present, via the reset instruction.
  - **Unlocated variables**
    On a warm start the unlocated variable, set via the reset instruction, maintains its present value (storing behavior).

NOTE: Should a buffered located variable/direct address be required, please use the RS or SR function blocks from the block library IEC.

Boolean AND (AND, AND (), ANDN, ANDN ())

AND Description

With AND a logical AND link occurs between the accu contents and the operand.

For the data types BYTE and WORD the link is made by bit.

AND Example

In the example D is "1", if A, B and C are "1".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`AND` `B`	The accu contents are AND-linked with the contents of "B".
`AND` `C`	The accu contents (result of the AND link from "A" and "B") are AND-linked with the contents of "C".
`ST D`	The link result is saved in "D".

AND () Description

AND can be used with the "(" left bracket modifier.

AND () Example

In the example D is "1", if A is "1" and B or C are "1".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`AND (`	The AND link is deferred until the right bracket is reached.
`LD B`	The contents of "B" are downloaded onto the accu.
`OR C`	The contents of "C" are OR-linked with the accu contents.
`)`	The deferred AND link is solved. The accu contents (result of the OR link from "B" and "C") are AND-linked with the contents of "A".
`ST D`	The link result is saved in "D".

ANDN Description

AND can be used with the N modifier.

ANDN Example

In the example D is "1", if A is "1" and B and C are "0".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`ANDN` `B`	The contents of "B" are inverted and AND-linked with the accu contents.
`ANDN` `C`	The contents of "C" are inverted and AND-linked with the accu contents (Result of the AND link from "A" and "B").
`ST D`	The link result is saved in "D".

ANDN () Description

AND can be used with the N modifier and the "(" left bracket modifier.

ANDN () Example

In the example D is "1", if A is "1", B is "0" and C is "1".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`ANDN (`	The AND link is deferred until the right bracket is reached.
`LD B`	The contents of "B" are downloaded onto the accu.
`ORN C`	The contents of "C" are inverted and OR-linked with the accu contents.
`)`	The deferred AND link is solved. The contents of "A" are inverted and AND-linked with the accu contents (Result of the OR link from "B" and "C").
`ST D`	The link result is saved in "D".

Boolean OR (OR, OR (), ORN, ORN ())

OR Description

With OR a logical OR link occurs between the accu contents and the operand.

For the data types BYTE and WORD the link is made by bit.

OR Example

In the example D is "1", if A or B is "1" and C is "1".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`OR` `B`	The accu contents are OR-linked with the contents of "B".
`AND C`	The accu contents (result of the AND link from "A" and "B") are AND-linked.
`ST D`	The link result is saved in "D".

OR () Description

OR can be used with the "(" left bracket modifier.

OR () Example

In the example D is "1", if A is "1" or B and C are "1".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`OR (`	The OR link is deferred until the right bracket is reached.
`LD B`	The contents of "B" are downloaded onto the accu.
`AND C`	The contents of "C" are AND-linked with the accu contents.
`)`	The deferred OR link is solved. The accu contents (Result of the AND link from "B" and "C") are OR linked with the contents of "A".
`ST D`	The link result is saved in "D".

ORN Description

ORN can be used with the N modifier.

ORN Example

In the example D is "1", if A is "1" or B is "0" and C is "1".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`ORN` `B`	The contents of "B" are inverted and OR linked with the accu contents.
`AND C`	The contents of "C" are AND linked with the accu contents (result of the OR link from "A" and "B").
`ST D`	The link result is saved in "D".

ORN () Description

ORN can be used with the N modifier and the "(" left bracket modifier.

ORN () Example

In the example D is "1", if A is "1" or B or C are "0".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`ORN (`	The OR link is deferred until the right bracket is reached.
`LD B`	The contents of "B" are downloaded onto the accu.
`AND C`	The contents of "C" are AND-linked with the accu contents.
`)`	The deferred OR link is solved. The accu contents (result of the AND link from "B" and "C") are OR linked with the contents of "A".
`ST D`	The link result is saved in "D".

Boolean exclusive OR (XOR, XOR (), XORN, XORN ())

XOR description

With XOR, a logical exclusive OR link is made between the accu contents and the operand.

If more than two operands are linked the result is "1" for an odd number of 1 conditions and "0" for an even number of 1 conditions.

For the data types BYTE and WORD the link is made by bit.

XOR example

In the example, D is "1" if A or B is "1". If A and B have the same status (both "0" or both "1"), D is "0".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`XOR` `B`	The accu contents are linked with the contents of the "B" exclusive OR.
`ST D`	The equation result is saved in "D".

XOR () description

XOR can be used with the "(" left bracket modifier.

XOR () example

In the example, D is "1" if A or the AND link from B and C is "1". If A and the result of the AND link have the same status (both "0" or both "1"), D is "0".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`XOR (`	The exclusive OR link is deferred until the right bracket is reached.
`LD B`	The contents of "B" are downloaded onto the accu.
`AND C`	The contents of "C" are AND-linked with the accu contents.
`)`	The reset exclusive OR link is performed. The accu contents (result of the AND link from "B" and "C") are exclusive OR-linked with the contents of "A".
`ST D`	The equation result is saved in "D".

XORN description

XOR can be used with the N modifier.

XORN example

In the example, D is "1" if A and B have the same contents (both "1" or both "0'). If A and B do not have the same status, D is "0".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`XORN` `B`	The contents of "B" are inverted and exclusive OR-linked with the accu contents.
`ST D`	The equation result is saved in "D".

XORN () description

XORN can be used with the "(" left bracket and N modifiers.

XORN () example

In the example, D is "1" if A and the AND link from B and C have the same contents (both "1" or both "0'). If A and B and the AND link from B and C do not have the same status, D is "0".

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accu.
`XORN (`	The exclusive OR link is deferred until the right bracket is reached.
`LD B`	The contents of "B" are downloaded onto the accu.
`AND C`	The contents of "C" are AND-linked with the accu contents.
`)`	The reset exclusive OR link is performed. The accu contents (result of the AND link from "B" and "C") are exclusive OR-linked with the contents of "A".
`ST D`	The equation result is saved in "D".

Invert (NOT)

NOT Description

The accumulator content is inverted with NOT.

NOT can only be used with Boolean data types (BIT, BYTE, WORD).

NOTE: This operator does not conform to IEC 61131-1.

Example NOT

Example NOT

Operation	Description
`LD A`	The contents of "A" are downloaded onto the accumulator.
`NOT`	The accumulator content is inverted.
`ST B`	The result is saved in "B".

Addition (ADD and ADD ())

ADD Description

With ADD the value of the operand is added to the accu contents.

ADD Example

The example corresponds to the formula D = A + B + C

Operation	Description
`LD A`	The value of "A" is downloaded onto the accu.
`ADD` `B`	The accu contents are added to the value of "B".
`ADD` `C`	The accu contents (sum of "A"+"B") are added to the value of "C".
`ST D`	The result is saved in "D".

ADD () Description

ADD can be used with the "(" left bracket modifier.

ADD () Example

The example corresponds to the formula D = A + (B - C)

Operation	Description
`LD A`	The value of "A" is downloaded onto the accu.
`ADD (`	The addition is deferred until the right bracket is reached.
`LD B`	The value of "B" is downloaded onto the accu.
`SUB C`	The value of "C" is subtracted from the accu contents.
`)`	The deferred addition is solved. The accu contents (result of "B"-"C") are added to the value of "A".
`ST D`	The result is saved in "D".

Subtraction (SUB and SUB ())

SUB Description

With SUB the value of an operand is subtracted from the accu contents.

SUB Example

The example corresponds to the formula D = A - B - C

Operation	Description
`LD A`	The value of "A" is downloaded onto the accu.
`SUB` `B`	The value of "B" is subtracted from the accu contents.
`SUB` `C`	The value of "C" is subtracted from the accu contents (result of "A"-"B").
`ST D`	The result is saved in "D".

Description SUB ()

SUB can be used with the "(" left bracket modifier.

Example SUB ()

The example corresponds to the formula D = A - (B - C)

Operation	Description
`LD A`	The value of "A" is downloaded onto the accu.
`SUB (`	The subtraction is reset until the right bracket is reached.
`LD B`	The value of "B" is downloaded onto the accu.
`SUB C`	The value of "C" is subtracted from the accu contents.
`)`	The reset subtraction is performed. The accu contents (result of "B"-"C") are subtracted from the value of "A".
`ST D`	The result is saved in "D".

Multiplication (MUL and MUL())

MUL Description

With MUL the accu contents are multiplied by the value of an operand.

MUL Example

The example corresponds to the formula D = A x B x C

Operation	Description
`LD A`	The value of "A" is downloaded onto the accu.
`MUL` `B`	The accu contents are multiplied by the value of "B".
`MUL` `C`	The accu contents (Result of "A"x"B") are multiplied by the value of "C".
`ST D`	The result is saved in "D".

Multiplication of TIME values

Normally the operand and the current accu contents must be of the same data type. The TIME data type in relation to MUL is an exception. In this case the accu content of data type TIME can be used together with an operand of data type ANY_NUM. After the execution of this instruction list the accu contents have, in this case, the data type TIME.

Example MUL with TIME values

The example corresponds to the formula t1 = t2 x i4.

Operation	Description
`LD t2`	The value of the time variables "t2" are downloaded onto the accu.
`MUL` `i4`	The accu contents are multiplied by the value of the integer variable "i4".
`ST t1`	The result is saved in the time variable "t1".

Description MUL ()

MUL can be used with the "(" left bracket modifier.

Example MUL ()

The example corresponds to the formula D = A x (B - C)

Operation	Description
`LD A`	The value of "A" is downloaded onto the accu.
`MUL (`	The multiplication is reset until the right bracket is reached.
`LD B`	The value of "B" is downloaded onto the accu.
`SUB C`	The value of "C" is subtracted from the accu contents.
`)`	The reset multiplication is performed. The accu contents (result of "B"-"C") are multiplied by the value of "A".
`ST D`	The result is saved in "D".

Division (DIV and DIV ())

DIV Description

With DIV the accu contents are divided by the value of an operand.

DIV example

The example corresponds to the formula D = A / B / C.

Operation	Description
`LD A`	The value of "A" is downloaded onto the accu.
`DIV` `B`	The accu contents are divided by the value of "B".
`DIV` `C`	The accu contents (result of "A"/"B") are divided by the value of "C".
`ST D`	The result is saved in "D".

Division of TIME values

Normally the operand and the current accu contents must be of the same data type. One exception is the data type TIME in connection with DIV. In this case the accu contents of data type TIME can be processed with an operand of data type ANY_NUM. After the execution of this instruction list the accu contents have, in this case, the data type TIME.

Example MUL with TIME values

The example corresponds to the formula t1 = t2 / i4.

Operation	Description
`LD t2`	The value of the time variables "t2" is downloaded onto the accu.
`DIV` `i4`	The accu contents are divided by the value of the integer variable "i4".
`ST t1`	The result is saved in the time variable "t1".

DIV () Description

DIV can be used with the "(" left bracket modifier.

DIV () Example

The example corresponds to the formula D = A / (B - C)

Operation	Description
`LD A`	The value of "A" is downloaded onto the accu.
`DIV (`	The division is reset until it the right bracket is reached.
`LD B`	The value of "B" is downloaded onto the accu.
`SUB C`	The value of "C" is subtracted from the accu contents.
`)`	The reset division is performed. The value of "A" is divided by the accu contents (result of "B"-"C").
`ST D`	The result is saved in "D".

Compare on "Greater Than" (GT and GT ())

Description GT

With GT the accu contents is compared with the operand contents. If the accu contents is greater than the operand contents, the result is a boolean "1". If the accu contents is less than or equal to the operand contents, the result is a boolean "0".

Example GT

Example GT

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`GT` `10`	The accu content is compared with the value ‘’0’’.
`ST D`	If the value of ‘’A’’ was less than ‘’10’’ (or equal ‘’10’’), the value ‘’0’’ is saved in ‘’D’’. If the value of ‘’A’’ was greater than ‘’10’’, the value ‘’1’’ is saved in ‘’D’’.

Description GT ()

GT can be used witht the modifier left bracket "(".

Examplel GT ()

Example GT ()

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`GT (`	The comparison is deferred until the right bracket has been reached.
`LD B`	The value of "B" is loaded into the accu.
`SUB C`	The value of "C" is subtracted from the accu content.
`)`	The deferred comparison is now carried out. The value of "A" is compared with the accu contents (result of "B"-"C").
`ST D`	If the value of "A" was less than "B"-"C" (or equal "B"-"C"), the value "0" is saved in "D". If the value of ‘’A’’ was greater than "B"-"C", the value ‘’1’’ is saved in ‘’D’’.

Compare to "Greater than/Equal to" (GE and GE ())

Description GE

With GE the accu contents is compared with the operand contents. If the accu contents is greater than or equal to the operand contents, the result is a Boolean "1". If the accu contents is less than the operand contents, the result is a Boolean "0".

GE example

E.g. GE

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`GE` `10`	The accu content is compared with the value ‘’10’’.
`ST D`	If the value of ‘’A’’ was less than ‘’10’’, the value ‘’0’’ is saved in ‘’D’’. If the value of ‘’A’’ was equal to or greater than ‘’10’’, the value ‘’1’’ is saved in ‘’D’’.

Description GT ()

GE can be used with the modifier left bracket "(".

GE () example

GE () example

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`GE (`	The comparison is deferred until the right bracket has been reached.
`LD B`	The value of "B" is loaded into the accu.
`SUB C`	The value of "C" is subtracted from the accu content.
`)`	The deferred comparison is now carried out. The value of "A" is compared with the accu contents (result of "B"-"C").
`ST D`	If the value of ‘’A’’ was less than "B"-"C", the value ‘’0’’ is saved in ‘’D’’. If the value of ‘’A’’ was equal to or greater than ‘’B’’ – "C", the value ‘’1’’ is saved in ‘’D’’.

Compare to "EQual to"(EQ and EQ ())

EQ description

With EQ the accu contents are compared with the operand contents. If the accu contents are equal to the operand contents, the result is a Boolean "1". If the accu contents are not equal to the operand contents, the result is a Boolean "0".

EQ example

EQ example

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`EQ` `10`	The accu contents are compared with the value ‘’10’’.
`ST D`	If the value of ‘’A’’ was not equal to ‘’10’’, the value ‘’0’’ is saved in ‘’D’’. If the value of ‘’A’’ was equal to ‘’10’’, the value ‘’1’’ is saved in ‘’D’’.

Description EQ ()

EQ can be used with the modifier left bracket "(".

EQ () example

EQ () example

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`EQ (`	The comparison is deferred until the right bracket has been reached.
`LD B`	The value of "B" is loaded into the accu.
`SUB C`	The value of "C" is subtracted from the accu content.
`)`	The deferred comparison is now carried out. The value of "A" is compared with the accu contents (result of "B"-"C").
`ST D`	If the value of ‘’A’’ was not equal to "B"-"C", the value ‘’0’’ is saved in ‘’D’’. If the value of ‘’A’’ was equal to "B"-"C", the value ‘’1’’ is saved in ‘’D’’.

Compare to "Not Equal to" (NE and NE ())

NE description

With NE the accu contents are compared with the operand contents. If the accu contents are not equal to the operand contents, the result is a Boolean "1". If the accu contents are equal to the operand contents, the result is a Boolean "0".

NE example

NE example

Command	Description
`LD A`	The value of "A" is loaded into the accu.
NE `10`	The accu contents are compared with the value ‘’10’’.
`ST D`	If the value of ‘’A’’ was equal to ‘’10’’, the value ‘’0’’ is saved in ‘’D’’. If the value of ‘’A’’ was not equal to ‘’10’’, the value ‘’1’’ is saved in ‘’D’’.

Description NE ()

NE can be used with the modifier left bracket "(".

NE () example

NE () example

Command	Description
`LD A`	The value of "A" is loaded into the accu.
NE (	The comparison is deferred until the right bracket has been reached..
`LD B`	The value of "B" is loaded into the accu.
`SUB C`	The value of "C" is subtracted from the accu content.
)	The deferred comparison is now carried out. The value of "A" is compared with the accu contents (result of "B"-"C").
`ST D`	If the value of ‘’A’’ was equal to "B"-"C", the value ‘’0’’ is saved in ‘’D’’. If the value of ‘’A’’ was not equal to "B"-"C", the value ‘’1’’ is saved in ‘’D’’.

Compare to "Less than/Equal to" (LE and LE ())

Description

With LE the accu contents are compared with the operand contents. If the accu contents are less than or equal to the operand contents, the result is a Boolean "1". If the accu contents are greater than the operand contents, the result is a Boolean "0".

LE example

LE example

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`LE` `10`	The accu contents are compared with the value ‘’10’’.
`ST D`	If the value of ‘’A’’ was greater than ‘’10’’, the value ‘’0’’ is saved in ‘’D’’. If the value of ‘’A’’ was less than or equal to ‘’10’’, the value ‘’1’’ is saved in ‘’D’’.

Description LE ()

LE can be used with the modifier left bracket "(".

LE () example

LE () example

Command	Description
`LD A`	The value of "A" is loaded into the accu.
`LE (`	The comparison is deferred until the right bracket has been reached.
`LD B`	The value of "B" is loaded into the accu.
`SUB C`	The value of "C" is subtracted from the accu content.
`)`	The deferred comparison is now carried out. The value of "A" is compared with the accu contents (result of "B"-"C").
`ST D`	If the value of ‘’A’’ was greater than "B"-"C", the value ‘’0’’ is saved in ‘’D’’. If the value of "A" was less than "B"-"C" (or equal to "B"-"C"), the value "1" is saved in "D".

Compare to "Less Than"(LT and LT ())

LT description

With LT the accu contents are compared with the operand contents. If the accu contents are less than the operand contents, the result is a Boolean "1". If the accu contents are greater than or equal to the operand contents, the result is a Boolean "0".

LT example

LT example

Command	Description
`LD A`	The value of "A" is loaded into the accu.
LT `10`	The accu contents are compared with the value ‘’10’’.
`ST D`	If the value of ‘’A’’ was greater than ‘’10’’ (or equal to ‘’10’’), the value ‘’0’’ is saved in ‘’D’’. If the value of ‘’A’’ was less than ‘’10’’, the value ‘’1’’ is saved in ‘’D’’.

Description LT ()

LT can be used with the modifier left bracket "(".

LT () example

LT () example

Command	Description
`LD A`	The value of "A" is loaded into the accu.
LT(	The comparison is deferred until the right bracket has been reached.
`LD B`	The value of "B" is loaded into the accu.
`SUB C`	The value of "C" is subtracted from the accu content.
)	The deferred comparison is now carried out. The value of "A" is compared with the accu contents (result of "B"-"C").
`ST D`	If the value of ‘’A’’ was greater than ‘’B’’-"C" (or equal to ‘’B’’-"C"), the value ‘’0’’ is saved in ‘’D’’. If the value of ‘’A’’ was less than "B"-"C", the value ‘’1’’ is saved in ‘’D’’.

Jump to label (JMP, JMPC and JMPCN)

JMP Description

With JMP a conditional or unconditional jump to a label is solved.

The label is used as an address and identifies the destination instruction. The destination instruction can be above or below the jump instruction. A label must always be the first element of a line. The label (max. 32 characters) must not be duplicated anywhere else in the project and there is no case sensitivity. The labels are separated by a colon ":" from the following instruction. Labels should only be at the beginning of "expressions", since otherwise an undefined value can be in the accu.

JMP Example

In the example an unconditional jump to the label "start" is solved.

Operation	Description
`start:  LD A`	The value of "A" is downloaded onto the accu.
`AND B`	Logical AND link between the accu contents and the contents of "B".
`OR C`	Logical OR link between the accu contents and the contents of "C".
`ST D`	The result of the links is saved in "D".
`JMP` `start`	Independent of the accu contents (value of "D") a jump to label "start" is solved.

JMPC and JMPCN Description

JMP can be used with the modifiers C and CN (only if the operand is of data type ANY_BIT).

JMPC Example

In the example a conditional jump (with "1") to label "start" is solved.

Operation	Description
`start:  LD A`	The value of "A" is downloaded onto the accu.
`AND B`	Logical AND link between the accu contents and the contents of "B".
`OR C`	Logical OR link between the accu contents and the contents of "C".
`ST D`	The result of the links is saved in "D".
`JMPC` `start`	The jump is only solved if the accu contents (value of "D") has the value "1".

JMPCN Example

In the example a conditional jump (by "0") to label "start" is solved.

Operation	Description
`start:  LD A`	The value of "A" is downloaded onto the accu.
`AND B`	Logical AND link between the accu contents and the contents of "B".
`OR C`	Logical OR link between the accu contents and the contents of "C".
`ST D`	The result of the links is saved in "D".
`JMPCN` `start`	The jump is only solved if the accu contents (value of "D") has the value "0".

Addresses

Possible addresses are:

- each LD instruction (see start1)
- each CAL instruction (see start2)
- the end of an instruction list (see start3)

Jumps cannot be made into other sections.

Example for possible addresses:

Operation	Description

`VAR`
`Timer_1 : TON;`
`END_VAR`
	Declaration of the function blocks TON.

`        LD IN1_BOOL`	
`        ST OT1_BOOL`	
`        JMPC start1`	Jump to start1, if OT1_BOOL = 1
`        LD IN1_BOOL`	
`        AND IN2_BOOL`	
`        JMPCN start2`	Jump to start2, if OT1_BOOL = 0
`        ST OT2_BOOL`	
`start1: LD IN1_INT`	
`        ADD IN2_INT`	
`        ST OT1_INT`	
`        JMP start3`	Unconditional jump after start3, JMPC/JMPCN is not allowed here as the accu contents are not of type BOOL.
`start2: CAL Timer_1 (IN:=IN3_BOOL, PT:=t#6s)`	
`        LD Timer_1.ET`	
`        ST OT1_TIME`	
`        LD Timer_1.Q`	
`        ST OT3_BOOL`	
`start3`	

Call Function Block/DFB (CAL, CALC and CALCN)

CAL Description

With CAL a function block or a DFB is conditionally or unconditionally called.

CALC and CALCN Description

CAL can be used with the Modifiers C and CN (only if the operand is of data type ANY_BIT).

Use of Function Blocks and DFBs

Use of Function Blocks and DFBs, page 372

FUNCNAME

Description

A function is performed with the function name (see *Function call, page 379*).

Right parenthesis ")"

At a Glance

With the right parenthesis ")" the editing of the reset operator is started. The number of right parenthesis operations must be equal to the number of left bracket modifiers. Brackets can be nested.

Example

In the example E will be "1", if C and/or D is "1", just as A and B are "1".

```
LD A
AND B
AND( C
OR D
)
ST E
```

10.4 Call up of functions, Function Blocks (EFBs) and Derived Function Blocks (DFBs)

Overview

This section describes the call up of functions, Function Blocks (EFBs) and Derived Function Blocks (DFBs).

What's in this Section?

This section contains the following topics:

Topic	Page
Use of Function Blocks and DFBs	372
Invoking a Function Block/DFB	374
Function call	379

Use of Function Blocks and DFBs

Use of Function Blocks and DFBs

Function blocks are provided by Concept in the form of libraries. The function block logic is created in C++ programming language and cannot be altered in the IL Editor. The names of the available function blocks can be taken from the block libraries.

DFBs are function blocks, which have been defined in Concept-DFB. There is no difference between functions and function blocks for DFBs. They are always handled as function blocks regardless of their internal structure.

The use of function blocks and DFBs consists of three parts in IL:
- the declaration *(see page 373)*,
- the function block/DFB invocation *(see page 374)*,
- the use of the function block/DFB outputs *(see page 373)*.

NOTE: The declaration of the function block/DFB invocation can take place manually or you can create the block end and the assignment of the parameters using the menu command **Objects** → **Insert FFB**.

Function Blocks with Limited Use

Use of the following EFBs from the DIAGNO block library is limited in IL (the function blocks can be used, but the expanded diagnostic information cannot be evaluated):
- XACT, XACT_DIA,
- XDYN_DIA,
- XGRP_DIA,
- XLOCK,
- XPRE_DIA,
- XLOCK_DIA,
- XREA_DIA

Function Blocks with Limited Invocation

For EFBs which have one or more outputs with data type ANY but no inputs with data type ANY (generic outputs/inputs), the block invocation can only take place in compact form *(see page 376)*. e.g. in the block library **LIB984**:
- GET_3X
- GET_4X

Unusable Function Blocks

Unusable Function Blocks:
- EFBs which use several registers with only the entry for the first register on the input/output (e.g. MBP_MSTR from the COMM block library) cannot be used.
- EFBs, which contain outputs with input information (e.g. GET_BIT, R2T from the LIB984 block library) cannot be used

- The following EFBs from the **COMM** block library cannot be used for the technical reasons listed above:
  - CREADREG
  - CREAD_REG
  - CWRITREG
  - CWRITE_REG
  - READREG
  - READ_REG
  - WRITEREG
  - WRITE_REG
  - MBP_MSTR
- The following EFBs from the **LIB984** block library cannot be used for the technical reasons listed above:
  - FIFO
  - GET_BIT
  - IEC_BMDI
  - LIFO
  - R2T
  - SET_BIT
  - SRCH
  - T2T

Declaration

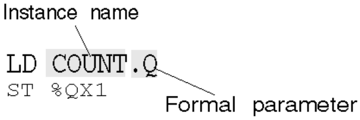
Before invoking the function block/DFBs, they must be declared using VAR and END_VAR *(see page 336)*.

Function Block/DFB Invocation

Invoking a Function Block/DFB, page 374

Use of the Function Block/DFB Outputs

The outputs of the function block/DFBs can always be used when a variable (read only) can also be used.



```
LD COUNT.Q
ST %QX1
```

Invoking a Function Block/DFB

At a Glance

The invocation can be made in 4 forms:

- using CAL with a list of the input parameters *(see page 374)*,
- using CAL with a list of the input/output parameters (compact form) *(see page 376)*,
- using CAL and Load/Save the input parameters *(see page 376)*,
- when using the input operators *(see page 377)*.

NOTE: Even if the function block has no inputs or the input parameters are not to be defined, the function block should be invoked (`CAL EFB_XY ()`) before the outputs can be used. Otherwise the initial values for the outputs are given, i.e. "0".

NOTE: In IL, unlike the graphic programming languages (FBD, LD), FB/DFB instances can be invoked multiple times.

CAL with a list of the input parameters

Function blocks/DFBs can be invoked using an instruction consisting of the CAL instruction followed by the instance names for the FBs/DFBs and a list, in brackets, of value assignments (current parameters) to formal parameters. The order of the formal parameters in a function block invocation is not significant. The list of the current parameters can be broken straight after a comma. It is not necessary for all formal parameters to be assigned a value. If a formal parameter is not assigned a value, the initial value defined in the variable editor is used when executing the function block. If an initial value is not defined, the default value (0) is used.

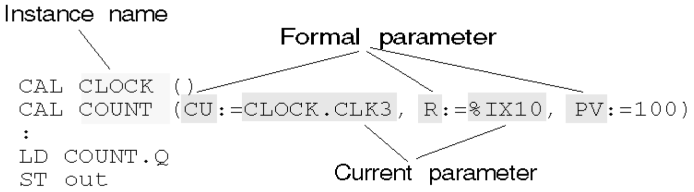
NOTE: Inputs of type VARINOUT *(see page 493)* always have to be assigned a value.

Using the CAL (..) instruction, setting the parameters for the function blocks/DFBs is ended. Then no more values can be sent to the FB/DFB. Only the output values can be read.

Example

CAL with a list of the input parameters

```
VAR
   CLOCK : SYSCLOCK ;
   COUNT : CTU_DINT ;
END_VAR
```

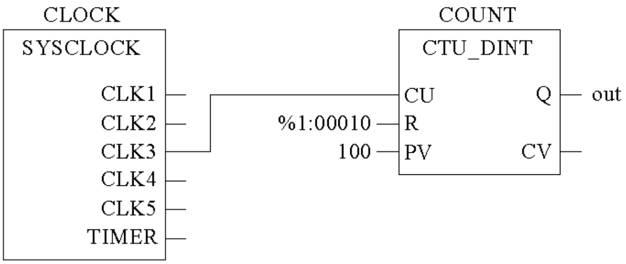


```
 CAL CLOCK ()
 CAL COUNT (CU:=CLOCK.CLK3, R:=%IX10, PV:=100)
 :
 LD COUNT.Q
 ST out
```

or

```
VAR
   CLOCK : SYSCLOCK ;
   COUNT : CTU_DINT ;
   Pulse : TP ;
END_VAR
CAL CLOCK ()
CAL COUNT(
   CU:=CLOCK.CLK3,
   R:=reset,
   PV:=100)
:
LD COUNT.Q
ST out
```

Invocation of the function block in FBD.



CAL with a list of the input/output parameters (compact form)

Block invocation and the assignments for the inputs/outputs are also possible in a more compact form, which saves runtime:

```
VAR
   CLOCK : SYSCLOCK ;
   COUNT : CTU_DINT ;
END_VAR

CAL CLOCK () ;
CAL COUNT (CU:=CLOCK.CLK3, R:=%IX10, PV:=100, Q=>out)
```

CAL with Loading/Saving of Input Parameters

Function blocks/DFBs can be invoked using an instruction list, which consists of loading the current parameters, followed by saving them in the formal parameters, followed by the CAL instruction. The order in which the parameters are loaded and saved is not significant. The list of the current parameters can be broken directly after a comma. It is not necessary for all formal parameters to be assigned a value. If a formal parameter is not assigned a value, the initial value defined in the variable editor is used when executing the function block. If an initial value is not defined, the default value (0) is used.

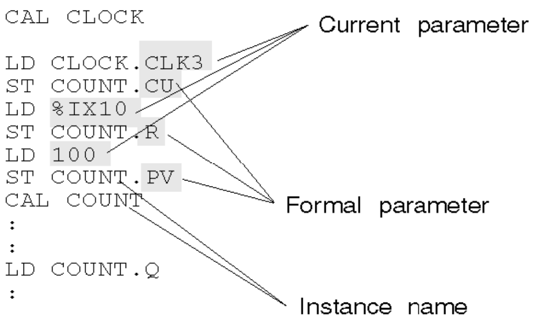
NOTE: Inputs of type VARINOUT *(see page 493)* always have to be assigned a value.

Using the CAL FBNAME instruction, setting the parameters for the function blocks/DFBs is ended. Then no more values can be sent to the FB/DFB. Only the output values can be read.

Only load and save instructions for the current FB/DFBs to be configured are allowed to be between the first load instruction for the current parameters and invocation of the function block/DFBs. All other instructions are not allowed in this position.

Example

CAL with Loading/Saving of Input Parameters



```
CAL CLOCK

LD CLOCK.CLK3
ST COUNT.CU
LD %IX10
ST COUNT.R
LD 100
ST COUNT.PV
CAL COUNT
:
:
LD COUNT.Q
:
```

Using the Input Operators

Function blocks can be called using an instruction list which consists of loading the current parameters, followed by saving them in the formal parameters, followed by an input operator. The order in which the parameters are loaded and saved is not significant. The list of the current parameters can be broken directly after a comma. It is not necessary for all formal parameters to be assigned a value. If a formal parameter is not assigned a value, the initial value defined in the variable editor is used when executing the function block. If an initial value is not defined, the default value (0) is used.

NOTE: Inputs of type VARINOUT *(see page 493)* always have to be assigned a value.

The possible input operators for the different function blocks can be taken from the table. Other input operators are not available.

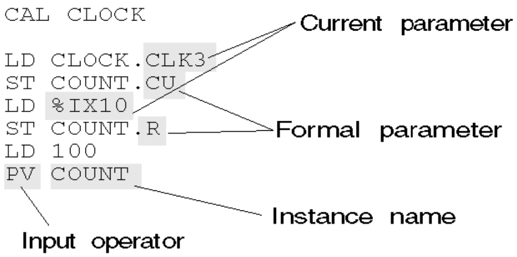
Input operator	FB Type
S1, R	SR
S, R1	RS
CLK	R_TRIG
CLK	F_TRIG
CU, R, PV	CTU_INT, CTU_DINT, CTU_UINT, CTU_UDINT
CD, LD, PV	CTD_INT, CTD_DINT, CTD_UINT, CTD_UDINT
CU, CD, R, LD, PV	CTUD_INT, CTUD_DINT, CTUD_UINT, CTUD_UDINT
IN, PT	TP
IN, PT	TON
IN, PT	TOF

Using input operator invocation, setting the parameters for the function blocks is ended. Then no more values can be sent to the FB. Only the output values can be read.

Only load and save instructions for the current FB being configured are allowed to be between the first load instruction for the current parameters and the input operator for the function block. All other instructions are not allowed in this position.

Example

Using the Input Operators



```
CAL CLOCK

LD CLOCK.CLK3
ST COUNT.CU
LD %IX10
ST COUNT.R
LD 100
PV COUNT
```

Function call

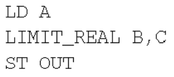
Use of functions

Functions are provided by Concept in the form of libraries. The logic of the functions is created in the programming language C++ and cannot be changed in the IL editor. You will find the names of the available functions in the block libraries.

Functions are called using an instruction list consisting of loading the **first** actual parameter into the battery and the name of the function. If necessary, this will be followed by a list of further actual parameters. The sequence in which the formal parameters in a function call are enumerated is not significant. Immediately following a comma, the list of the actual parameters may be wrapped. The result of the function becomes the battery content after the function has been executed, and can be saved using ST *(see page 341)* in an operand, or may directly be processed further.

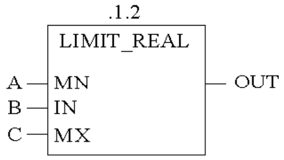
NOTE: The declaration of function calls may either be generated manually, or you may generate the block rump and the allocation of the parameters using the menu command **Objects →Insert FFB.**

The picture shows calling a function in IL.



```
LD A
LIMIT_REAL B,C
ST OUT
```

The picture shows calling a function in FDP.

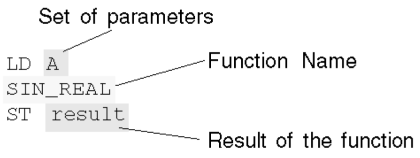


Functions which cannot be used

Functions having one or more outputs of data type ANY but no inputs of data type ANY (generic outputs/inputs) cannot be used in IL.

Calling a function with an input

If the function to be executed has only got one input, the name of the function is not followed by a list of actual parameters.

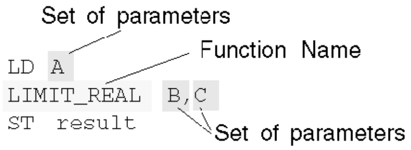


```
LD A
SIN_REAL
ST result
```

Calling a function with multiple inputs

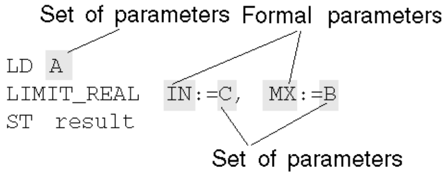
If the function to be executed has several inputs, there are two possibilities for assigning the actual parameters.

- The name of the function is followed by a list of the actual parameters



```
LD A
LIMIT_REAL  B,C
ST  result
```

- The name of the function is followed by a list of the value assignments (actual parameters) to the formal parameters.



```
LD A
LIMIT_REAL  IN:=C,  MX:=B
ST  result
```

Function calls including processing of the battery value

If the value to be processed is already in the battery, it is not necessary to use the loading instruction.

```
LIMIT_REAL B,C
ST result
```

Function calls including further direct processing of the result

If the result is immediately to be processed further it is not necessary to include the memory instruction.

```
LD A
LIMIT_REAL B,C
MUL E
```

10.5 Syntax check and Code generation

Overview

This section describes the syntax check and the code generation with the IL instruction list.

What's in this Section?

This section contains the following topics:

Topic	Page
Syntax Check	382
Code generation	384

Syntax Check

Introduction

A syntax check can be performed during the program/DFB creation with **Project → Analyze section**.

Syntax Check Options

With the menu command **Options →Preferences →IEC extensions... →IEC-extensions** the syntax check options can be defined.

NOTE: The settings in this dialog are used in the project description (PRJ.DSK) and in the Concept installations description (CONCEPT.DSK), i.e. they are valid for the entire Concept installation.

If a project is opened, that was created using different settings (e.g. **Allow nested comments** in the project and not in the actual Concept Installation), errors can occur when opening the project.

Allow Case Insensitive Keywords

If the check box **Allow case insensitive keywords** is checked, upper and lower case for all keywords is enabled.

Allow nested comments

If the check box **Nested comments authorized** is checked, nested comments can be entered. There are no limits to the nesting depths.

Comments everywhere in the text permitted (IL)

If the check box **Comments everywhere in the text permitted (IL)** is checked, comments can be placed anywhere in the IL section.

Additional Variable Names Permitted (IL)

If the check box **Additional variable names permitted (IL)** is checked, the use of additional variable names (e.g. "S1" or "IN") is possible in IL. (These variables can always be used in FBD, LD and ST.)

Allow Leading Digits in Identifiers

If the check box **Allow leading digits in identifiers** is checked, figures as the first character of identifiers (i.e. variable names, step names, EFB names) are possible. Identifiers, which consist solely of figures are, however, not authorized, they must contain at least one letter.

Unused Parameters Cause Warnings

The IEC 1131-3 permits functions and Function Blocks to be called up without calling up the assignment of all the input parameters. These unused parameters are implicitly assigned a 0, or they retain the value from the last call up (Function Blocks only).

If in the menu command **Options** →**Preferences** →**Analysis...** →**Analysis** the check box **Unused parameters lead to warnings** is checked, a list of these unused parameters is displayed in the message window when generating the code.

Code generation

At a Glance

The menu command **Project →Options for code generation** is used to define options for code generation.

Fastest code (restricted check)

If the check box **Fastest Code (restricted check)** is activated, a runtime-optimized code will be generated.

This runtime optimization is achieved with integer arithmetic (e.g. "+" or "-") with simple process commands instead of EFB calls.

Process commands are much faster than EFB calls, but they generate no error messages, such as e.g. Arithmetic or Array overrun. This option should only be used if it has been ensured that the program is free of arithmetic errors.

Example: Fastest Code

```
LD in1
ADD 1
ST out1
```

If **Fastest Code (restricted check)** is selected, the addition "in1 + 1" is executed with the process command "add". The code is now faster than if EFB ADD_INT had been called up. However, no runtime error is generated if "in1" is 32767. In this case, "out1" would overrun from 32767 to -32768!

Activate loop control

This check box activates a software watchdog for continuous loops.

If this check box is checked, with loops within IL and ST sections, it is tested whether these loops are again exited within a certain time. The time authorized depends on the manually defined watchdog time. The authorized time for **all loops** combined constitutes 80% of the Hardware watchdog time. In this way triggering of the hardware watchdog by endless loops is disabled. If a time consuming loop or an endless loop is detected, processing of the affected section will stop, an entry in the Event display will be generated and processing of the next section will begin. In the next cycle, the segment will be re-processed until a time consuming loop or an endless loop is detected once again, or until the segment is completed correctly.

NOTE: If the hardware watchdog stops the PLC when a time consuming loop or an endless loop is detected, this option should not be activated. The hardware watchdog itself is not switched off by this function.

10.6 Online functions of the IL instruction list

Overview

This section describes the online functions of the IL instruction list.

What's in this Section?

This section contains the following topics:

Topic	Page
Animation	386
Monitoring field	389

Animation

At a Glance

There are two animation modes available in the IL and ST editor:

- Animation of binary variables
- Animation of selected variables

Animation of binary variables

The animation of the selected objects is activated with the menu command **Online →Animate selection**.

In this mode, the current signal status of binary values is shown in the editor window.

The animation of direct addresses and direct FB input/outputs is not possible.

Animation of selected variables

The dialog box for the display of the current signal status of selected variables is activated with the menu command **Online →View selected**.

Furthermore, at least one variable, which can be animated, must be selected.

Variables and multi-element variables that can be selected are denoted by red, green or yellow script.

Properties of the dialog box

The name of the selected variables or multi-element variables are shown in the dialog box, with the data type and current value.

The dialog box is modeless, i.e. it remains open until it is closed or the animation is terminated. If several text language sections are open and clicked on in this dialog box, a dialog box is opened for each section. The name of the section is displayed in the dialog box heading.

Color key

There are 12 different color schemes available for animation. An overview of the color scheme and the meaning of each color can be found in the Online help (Tip: Search the online help for the index reference "Colors").

Inserting several variables

The procedure for inserting several variables is as follows:

Step	Action
1	Select the desired variables or multi-element variables.
2	Accept this with **Online →Animate selected** in the dialog box.

Inserting all variables

The procedure for inserting all the variables is as follows:

Step	Action
1	Select the whole section with **CTRL**+**A**.
2	Migrate all variables and multi-element variables of the dialog section with **Online** →**Animate selcted** to the dialog box.

Altering column width

The procedure for altering the column width is as follows:

Step	Action
1	Position the mouse pointer on the right margin button. **Reaction:** The mouse pointer changes its shape to ◂‖▸.
2	Alter the column width by dragging with the left mouse button depressed.

Multi-element variables

With multi-element variables the display of the elements can be switched on or off.

Action	Function	Condition
Click on symbol **+** or key **+**	The next component level for the current line is shown.	When using the keyboard, the cursor must remain on a **+** symbol.
Key **x** (number lock)	All component levels for the current line are shown.	The cursor must remain on a **+** symbol.
Click on symbol **-** or key **-**	All component levels for the current line, which are shown, are grayed out.	When using the keyboard, the cursor must remain on a **-** symbol.
CTRL+**+**	The display of the components of the current line is restored (Restoration of display before the last activation of -	The cursor must remain on a **+** symbol.
CTRL+**x** (number lock)	All component levels of the current multi-element variables are shown.	The cursor must remain on an element of a multi-element variable.
CTRL+**-**	All component levels of the current multi-element variables are grayed out.	The cursor must remain on an element of a multi-element variable.
CTRL+**end**	to go to the end of the table	
CTRL+**Pos1**	to go to the start of the table	

Saving and restoring animation

With the menu command **Save animation** the settings (e.g. Position of monitoring fields) of the current animation can be saved. After terminating this animation, the animation can be restored with the same settings via the menu command **Restore animation**.

NOTE: To avoid inconsistencies between the program on the PC and the PLC and to also have the animation available in the next Concept sitting, the project must be saved when quitting Concept

Monitoring field

At a Glance

With the menu command **Online →Selected in Inspect field** a monitoring field can be entered in the section. The current value of the assigned variables is shown in this monitoring field.

Limitations

The generation of monitoring fields for direct addresses and direct FB input/outputs (INST.Q) is not possible.

Display of multi-element variables

With multi-element variables, the value of the first element is shown.

If a view of more elements is desired, this can be defined in the dialog **Settings for monitoring field**, which is called up by double clicking on the monitoring field.

Minimum and maximum values

In the dialog **Settings for monitoring field**, which can be called up with a double click on the monitoring field, a minimum and maximum value can be defined for the monitored variable. If the variable violates one of these thresholds, this will be displayed in color in the monitoring field.

An overview of the color scheme and the meaning of each color can be found in the Online help (Tip: Search the online help for the index reference "Colors").

Generating a monitoring field

The procedure for generating a monitoring field is as follows:

Step	Action
1	Select a variable (e.g. double-click on variable).
2	Execute the menu command **Online →Selected in Inspect field**. **Reaction:** The section animation is started (gray section background) and the cursor symbol changes into box symbol.
3	Position the cursor to any position in the section and click with the left mouse button. **Reaction:** A monitoring field, consisting of variable name and value, is generated for the selected variable on the chosen position.

10.7 Creating a program with the IL instruction list

Creating a program in the IL instruction list.

At a Glance

The following description contains an example of creating a program in IL instruction list. The creation of a program in IL instruction list is organized into 2 main steps:

Step	Action
1	Generating a section *(see page 390)*
2	Creating the logic *(see page 391)*

Generating a section

The procedure for generating a section is as follows:

Step	Action
1	Using the menu command **File** →**New section...** and enter a section name. **Note:** The section name (max. 32 characters) must be clear throughout the project, so that there is no difference regarding case sensitivity. If the name entered already exists, a warning is given and another name must be chosen. The section name must correspond to the IEC name conventions, otherwise an error message arises. **Note:** In accordance with IEC1131-3, only letters are permitted as the first character of names. Should numbers be required as the first character, however, the menu command **Options** →**Presettings** →**IEC expansions...** →**Enable leading figures in identifiers** .

Creating the logic

The procedure for creating the logic is as follows:

Step	Action
1	Declare the Function Block and DFBs, which are to be used, with assistance from VAR…END_VAR. **Example:** VAR RAMP_UP, RAMP_DOWN, RAMP_X : TON ; COUNT : CTU_DINT ; END_VAR
2	Declare the variables and their initial value in the Variable Editor.
3	Create the logic of the program. **Example:** LD A SIN_REAL MUL_REAL B,C ST D LD Y AND X JMPC endl LD M SIN_REAL MUL_REAL N,O ST P JMP end2 end1: LD D ST %QD4 end2: LD P ST %QD5
4	save the section with the menu command **Data file →Save project** .

Structured text ST

11

Overview

This Chapter describes the programming language structured text ST which conforms to IEC 1131.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
11.1	General information about structured Text ST	394
11.2	Expressions	396
11.3	Operators of the programming language of structured ST text	402
11.4	Assign instructions	422
11.5	Call up of functions, Function Blocks (EFBs) and Derived Function Blocks (DFBs)	441
11.6	Syntax check and code generation	448
11.7	Online functions of the ST programming language	451
11.8	Creating a program with the structured ST text	452

11.1 General information about structured Text ST

General Information about the ST Structured Text

Introduction

With the programming language of structured text (ST), it is possible, for example, to call up Function Blocks, perform functions and assignments, conditionally perform instructions and repeat tasks.

Spell Check

Spelling is immediately checked when key words, separators and comments are entered. If a key word, separator or comment is recognized, it is identified with a color surround. If unauthorized key words (instructions or operators) are entered, it is likewise identified in color.

IEC Conventions

The IEC 1131 does not permit the input of direct addresses in the usual Concept form. To input direct addresses see *Operands, page 397*.

In accordance with IEC 113-3, key words must be entered in upper case. Should the use of lower case letters be required, they can be enabled in the dialog box **Options →Preferences →IEC Extensions... →IEC expansions** with the option **Allow case insensitive keywords**.

Blank spaces and tabs have no influence upon the syntax and can be used freely.

Context help

With the right mouse button an object can be selected and at the same time a context sensitive menu called up. Therefore, for example, with FFBs the right mouse button can call up the associated block description.

Syntax Check

A syntax check can be performed during the program/DFB creation with **Project → Analyze section**, see also *Syntax Check, page 449*.

Codegeneration

Using the **Project →Code Generation Options** menu command, you can define options for code generation, see also *Code generation, page 450*.

Editing with the Keyboard

Normally editing in Concept is performed with the mouse, however it is also possible with the keyboard (see also *Short Cut Keys in the IL, ST and Data Type Editor, page 834*).

IEC Conformity

For a description of the IEC conformity of the ST programming language see *IEC conformity, page 849*.

11.2 Expressions

Overview

This section contains an overview of the expressions in the programming language of structured text ST.

expressions consists of operands and operators.

What's in this Section?

This section contains the following topics:

Topic	Page
Operands	397
Operators	399

Operands

At a Glance

An operand can be:

- a literal,
- a variable,
- a multi-element variable,
- an element of a multi-element variable,
- a function call up,
- a FB/DFB output or
- a direct address.

Access to the field variables

When accessing field variables (ARRAY), only literals and variables of ANY_INT type are permitted in the index entry.

Example: Using field variables

```
var1[i] := 8 ;
var2.otto[4] := var3 ;
var4[1+i+j*5] := 4 ;
```

Type conversion

Data types, which are in an instruction of processing operands, must be identical. Should operands of various types be processed, a type conversion must be performed beforehand.

An exception is the data type TIME in conjunction with the arithmetic operators "*" (multiplication) and "/" (division). With both these operators, an operand of TIME data type can be processed together with an operand of ANY_NUM data type. The result of this instruction has in this instance the data type TIME.

Example: Integer variable and real variable

In the example the integer variable i1 is converted into a real variable before being added to the real variable r4.

```
r3 := r4 + SIN_REAL(INT_TO_REAL(i1)) ;
```

Example: Integer variable and time variable

In the example the time variable t2 is multiplied by the integer variable i4 and the result is stored in the time variable t1.

```
t1 := t2 * i4 ;
```

Default data types of direct addresses

The following table shows the default data types of direct addresses:

Input	Output	Default data type	possible data type
%IX,%I	%QX,%Q	BOOL	BOOL
%IB	%QB	BYTE	BYTE
%IW	%QW	INT	INT, UINT, WORD
%ID	%QD	REAL	REAL, DINT, UDINT, TIME

Using other data types

Should other data types be assigned as default data types of a direct address, this must be done through an explicit declaration (VAR…END_VAR *(see page 426)*). VAR…END_VAR cannot be used in Concept for the declaration of variables. The variable declaration is performed conveniently by using the Variable Editor *(see page 543)*.

Operators

Introduction

An operator is a symbol for:
- an arithmetic operation to be executed or
- a configured operation to be executed or
- the function call up.

Operators are generic, i.e. they are automatically matched with the operands data type.

NOTE: Operators can be either entered manually or generated with assistance from the menu **Objects** →**Operators**.

Expression Evaluation

The evaluation of an expression consists of applying the operators to the operands, in the sequence, which is defined by the order of the operators rank (see table). The operator with the highest rank in an expression is performed first, followed by the operator with the next highest rank etc. until the evaluation is complete. Operators with the same rank are performed from left to right, as they are written in the expression. This sequence can be altered with the use of parentheses.

Table of Operators

ST programming language operators:

Operator	Meaning	possible operand	Order of rank	see also
()	Use of parentheses:	Expression	1 (highest)	*Use of parentheses "()", page 403*
FUNCNAME (current parameter list)	Function editing (call up)	Expression, literal, variable, direct address of ANY data type	2	*Function Invocation, page 446*
-	Negation	Expression, literal, variable, direct address of ANY_NUM data type	3	*Negation (-), page 406*
NOT	Complement	Expression, literal, variable, direct address of ANY_BIT data type	3	*Complement formation (NOT), page 407*

Operator	Meaning	possible operand	Order of rank	see also
**	Exponentiation	Expression, literal, variable, direct address of REAL data type (basis), ANY_NUM (exponent)	4	*Exponentiation (**), page 405*
*	Multiplication	Expression, literal, variable, direct address of ANY_NUM data type or TIME data type	5	*Multiplication (*), page 408*
/	Division	Expression, literal, variable, direct address of ANY_NUM data type	5	*Division (/), page 409*
MOD	Modulo	Expression, literal, variable, direct address of ANY_INT data type	5	*Modulo (MOD), page 410*
+	Addition	Expression, literal, variable, direct address of ANY_NUM data type or TIME data type	6	*Addition (+), page 411*
-	Subtraction	Expression, literal, variable, direct address of ANY_NUM data type or TIME data type	6	*Subtraction (-), page 412*
<	Less-than comparison	Expression, literal, variable, direct address of ANY_ELEM data type	7	*Comparison with "Less Than"(<), page 417*
>	Greater-than comparison	Expression, literal, variable, direct address of ANY_ELEM data type	7	*Comparison on "Greater Than" (>), page 413*
<=	Less or equal to comparison	Expression, literal, variable, direct address of ANY_ELEM data type	7	*Comparison with "Less than or Equal to" (<=), page 418*
>=	Greater or equal to comparison	Expression, literal, variable, direct address of ANY_ELEM data type	7	*Comparison on "Greater than/Equal to" (>=), page 414*
=	Equality	Expression, literal, variable, direct address of ANY_ELEM data type	8	*Comparison with "EQual to" (=), page 415*
<>	Inequality	Expression, literal, variable, direct address of ANY_ELEM data type	8	*Comparison with "Not Equal to" (<>), page 416*

Operator	**Meaning**	**possible operand**	**Order of rank**	**see also**
&, AND	configured AND	Expression, literal, variable, direct address of ANY_BIT data type	9	*Boolean AND (AND or &), page 419*
XOR	Configured exclusive OR	Expression, literal, variable, direct address of ANY_BIT data type	10	*Boolean Exclusive OR (XOR), page 421*
OR	Configured OR	Expression, literal, variable, direct address of ANY_BIT data type	11 (lowest)	*Boolean OR (OR), page 420*

11.3 Operators of the programming language of structured ST text

Overview

This section describes the operators of the programming language of structured ST text.

What's in this Section?

This section contains the following topics:

Topic	Page
Use of parentheses "()"	403
FUNCNAME	404
Exponentiation (**)	405
Negation (-)	406
Complement formation (NOT)	407
Multiplication (*)	408
Division (/)	409
Modulo (MOD)	410
Addition (+)	411
Subtraction (-)	412
Comparison on "Greater Than" (>)	413
Comparison on "Greater than/Equal to" (>=)	414
Comparison with "EQual to" (=)	415
Comparison with "Not Equal to" (<>)	416
Comparison with "Less Than"(<)	417
Comparison with "Less than or Equal to" (<=)	418
Boolean AND (AND or &)	419
Boolean OR (OR)	420
Boolean Exclusive OR (XOR)	421

Use of parentheses "()"

Description

Brackets are used to alter the execution sequence of the operators.

Example of parentheses "()"

If the operands A, B, C, and D have the values "1", "2", "3", "and -4",

```
A+B-C*D
```

has the result 15 and

```
(A+B-C)*D
```

has the result 0.

FUNCNAME

Description

The function processing is used to perform functions (see *Function Invocation, page 446*).

Exponentiation (**)

Description

For exponentiation "**" the value of the first operand (basis) is potentiated with that of the second operand (exponent).

NOTE: Exponentiation operates in the ST programming languageand with a resolution of 23 bits. For the graphic languages the exponentiation operates with a resolution of 24 bits..

Example: Exponentiation "**"

In the example OUT will be"625.0", when IN1 is "5.0" and IN2 is "4.0".

```
OUT := IN1 ** IN2 ;
```

Negation (-)

Description

During negation "-" a sign reversal for the value of the operand takes place.

Example: Negation "-"

In the example OUT will be "-4", when IN1 is "4".

```
OUT := - IN1 ;
```

Complement formation (NOT)

Description

In NOT a bit by bit inversion of the operands takes place.

Example: NOT

In the example OUT will be"0011001100", when IN1 is "1100110011".

```
OUT := NOT IN1 ;
```

Multiplication (*)

Description

For multiplication "**" the value of the first operand is multiplied with that of the second operand (exponent).

Example: Multiplication "*"

```
OUT := IN1 * IN2 ;
```

Multiplication of TIME values

Normally the data types of the operands to be processed must be identical to an instruction. However, the multiplication forms an exception when combined with data type TIME. In this case an operand with the datentype TIME combined with an operanden of data type ANY_NUM can be processed. In this case the result of this instruction has the data type TIME.

Example: Multiplication of TIME values

In the example the Time variable t2 is multiplied by the integer variables i4 and the result is deposited in the t1 Time variables.

```
t1 := t2 * i4 ;
```

Division (/)

Description

For division "/" the value of the first operand is divided by that of the second operand (exponent).

Example: Division "/"

```
OUT := IN1 / IN2 ;
```

Division of TIME values

Normally the data types of the operands to be processed must be identical to an instruction. However the division forms an exception when combined with data type TIME. In this case an operand with the data type TIME combined with an operand of data type ANY_NUM can be processed. In this case the result of this instruction has the data type TIME.

Example division of TIME values

In the example the Time variable t2 is divided by the integer variables i4 and the result is deposited in the t1 Time variables.

```
t1 := t2 / i4 ;
```

Modulo (MOD)

Description

For MOD the value of the first operandis divided by that of the second operand and the remainder of thedivision (Modulo) is displayed as the result.

Example: MOD

```
OUT := IN1 MOD IN2 ;
```

Addition (+)

Description

For the addition "+" the value of the first operand is added to that of the second operand.

Example: Addition "+"

```
OUT := IN1 + IN2 ;
```

Subtraction (-)

Description

For the subtraction "-" the value of the second operand is subracted from that of the first operand.

Example: Subtraction "-"

```
OUT := IN1 - IN2 ;
```

Comparison on "Greater Than" (>)

Description

With ">" the value of the first operand is compared with that of the second operand. If the first operand is greater than the second, the result is a boolean "1". If the first operand is less than or equal to the second, the result is a Boolean "0".

Example: Greater Than ">"

In the example "OUT" will be "1" if "IN1" is greater than "10" and "0", if "IN1" is less than "0".

```
OUT := IN1 > 10 ;
```

Comparison on "Greater than/Equal to" (>=)

Description

With ">=" the value of the first operand is compared to that of the second operand. If the first operation is greater than or equal to the second, the result is a Boolean "1". If the first operand is less than the second, the result is a Boolean "0".

Example: Greater Than/Equal ">="

In the example "OUT"will be "1"if "IN1" is greater than/equal to "10" and otherwise "0".

```
OUT := IN1 >= 10 ;
```

Comparison with "EQual to" (=)

Description

The value of the first operation is compared with the value of the second with "=". If the first operation is equal to the second, the result is a Boolean "1". If the first operation is not equal to the second, the result is a Boolean "0".

Example: Equal "="

In the example, "OUT" will be "1", if "IN1" is equal to "10" – otherwise it will be "0".

```
OUT := IN1 = 10 ;
```

Comparison with "Not Equal to" (<>)

Description

The value of the first operation is compared with the value of the second with "<>". If the first operation is not equal to the second, the result is a Boolean "1". If the first operation is equal to the second, the result is a Boolean "0".

Example: Not Equal "<>"

In the example, "OUT" will be "1", if "IN1" is not equal to "10" – otherwise it will be "0".

```
OUT := IN1 <> 10 ;
```

Comparison with "Less Than"(<)

Description

The value of the first operation is compared with the value of the second with "<". If the first operation is smaller than the second, the result is a Boolean "1". If the first operation is bigger than or the same size as the second, the result is a Boolean "0".

Example: Less Than "<"

In the example, "OUT" will be "1", if "IN1" is less than "10" – otherwise it will be "0".

```
OUT := IN1 < 10 ;
```

Comparison with "Less than or Equal to" (<=)

Description

The value of the first operation is compared with the value of the second with "<=". If the first operation is less than or equal to the second, the result is a Boolean "1". If the first operation is greater than the second, the result is a Boolean "0".

Example: Less Than or Equal "<="

In the example, "OUT" will be "1", if "IN1" is less than or equal to "10" – otherwise it will be "0".

```
OUT := IN1 <= 10 ;
```

Boolean AND (AND or &)

Description

With "AND" or "&" a configured AND link occurs between the operations.

With the BYTE and WORD data types, the link is performed bit by bit.

Example: Boolean "AND or &"

In the examples, "OUT" will be "1" if "IN1", "IN2" and "IN3" are "1".

```
OUT := IN1 AND IN2 AND IN3 ;
```

or

```
OUT := IN1 AND IN2 AND IN3 ;
```

Boolean OR (OR)

Description

With OR, a configured OR link occurs between the operations.

With the BYTE and WORD data types, the link is performed bit by bit.

Example Boolean OR "OR"

In the example, "OUT" will be "1" if "IN1", "IN2" or "IN3" is "1".

```
OUT := IN1 OR IN2 OR IN3 ;
```

Boolean Exclusive OR (XOR)

Description

With XOR, a configured Exclusive OR link occurs between the operations.

With the BYTE and WORD data types, the link is performed bit by bit.

Example: Boolean Exclusive OR "XOR"

In the example "OUT" will be "1", if "IN1" and "IN2" are not equal. If "IN1" and "IN2" have the same state (both "0" or "1"), "OUT" is "0".

```
OUT := IN1 XOR IN2 ;
```

Linking more than 2 operations

If more than two operations are linked, the result is "1" with an odd number of 1-states and "0" with an even number of 1-states.

Example: Linking more than 2 operations

In the example, "OUT" will be "1" if 1, 3 or 5 operations are "1". "OUT" will be "0" if 0, 2 or 4 operations are "1".

```
OUT := IN1 XOR IN2 XOR IN3 XOR IN4 XOR IN5;
```

11.4 Assign instructions

Overview

This section describes the instructions for the programming language of structured ST text.

What's in this Section?

This section contains the following topics:

Topic	Page
Instructions	423
Assignment	424
Declaration (VAR...END_VAR)	426
IF...THEN...END_IF	428
ELSE	429
ELSIF...THEN	430
CASE...OF...END_CASE	431
FOR...TO...BY...DO...END_FOR	432
WHILE...DO...END_WHILE	435
REPEAT...UNTIL...END_REPEAT	437
EXIT	438
Empty instruction	439
Comment	440

Instructions

Description

Instructions are the "commands" of the ST programming language.

Instructions must be completed by semicolons. Several instructions can be entered in one line (separated by semicolons).

NOTE: Instructions can be either entered manually or generated using the menu **Objects**.

Assignment

At a Glance

When an assignment is performed, the current value of a single or multi-element variable is replaced by the result of the evaluation of the expression

An assignment consists of a variable specification on the left side, followed by the assignment operator ":=", followed by the expression to be evaluated. Both variables must be of the same data type.

Assigning the value of a variable to another variable

Assignments are used to assign the value of a variable to another variable.

The instruction

```
A := B ;
```

is for instance used to replace the value of the variable "A" by the current value of the variable "B". If "A" and "B" are of an elementary data type, the individual value "B" is passed to "A". If "A" and "B" are of a derived data type, the values of all elements are passed from "B" to "A".

Assigning the value of a literal to a variable

Assignments are used to assign a literal to variables.

The instruction

```
C := 25 ;
```

is for instance used to assign the value "25" to the variable "C".

Assigning the value of an FFB to a variable

Assignments are used to assign a value to a variable which is returned by a function or a function block.

The instruction

```
B := MOD_INT(C,A) ;
```

is for instance used to assign the modulo of the variables "C" and "A" to the variable "B".

The instruction

```
A := TON1.Q ;
```

is for instance used to assign to the variable "A" the value of the output "Q" of the function block TONI.

Assigning the value of an operation to a variable

Assignments are used to assign to a variable a value which is the result of an operation.

The instruction

```
X := (A+B-C)*D ;
```

is for instance used to assign to the variable "X" the result of the operation "(A+B-C)*D".

Declaration (VAR...END_VAR)

At a Glance

The VAR instruction is utilized for declaring the function blocks used and DFBs and declaring direct addresses if they are not to be used with the default data type. VAR cannot be used for declaring a variable in Concept. Declaring the variables may conveniently be done via the Variables editor.

The END_VAR instruction marks the end of the declaration.

NOTE: The declaration of the FBs/DFBs and direct addresses applies only to the current section. If the same FFB type or the same address are also used in another section, the FFB type or the address must be declared again in this section.

Declaration of function blocks and DFBs

Every time a FB/DFB example is used, a unique example name is assigned when it is declared. The example name is used to mark the function block uniquely in a project. The example name must be unique in the whole project; no distinction is made between upper/lower case. The example name must correspond to the IEC Name conventions, otherwise an error message will be displayed.

After specifying the example name, the function block type, e.g.CTD_DINT is specified.

In the case of function block types no data type is specified. It is determined by the data type of the actual parameters. If all actual parameters consist of literals, a suitable data type will be selected.

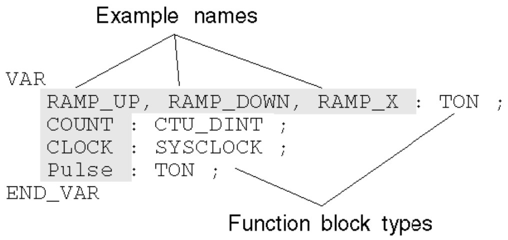
Any number of example names may be declared for an FB/DFB.

NOTE: The dialog **Objects →Insert FFB** provides you with a form for creating the FB-/DFB declaration in a simple and speedy manner.

NOTE: In contrast to grafic programming languages (FBD, LD), it is possible to execute multiple calls in FB/DFB examples within ST.

Example

Declaration of function blocks and DFBs



```
VAR
   RAMP_UP, RAMP_DOWN, RAMP_X : TON ;
   COUNT : CTU_DINT ;
   CLOCK : SYSCLOCK ;
   Pulse : TON ;
END_VAR
```

Declaration of direct addresses

In the case of this declaration, every direct address used whose data type does not correspond to the default data type will be assigned the required data type (see also Default data types of direct addresses *(see page 328)*).

Example

Declaration of direct addresses

```
VAR
   AT %QW1 : WORD ;
   AT %IW15 : UINT ;
   AT %ID45 : DINT ;
   AT %QD4 : TIME ;
END_VAR
```

IF...THEN...END_IF

Description

The IF instruction determines that an instruction or a group of instructions will only be executed if its related Boolean expression has the value 1 (true). If the condition is 0 (false), the instruction or the instruction group will not be executed.

The THEN-command identifies the end of the condition and the beginning of the command(s).

The END_IF instruction marks the end of the instruction(s).

NOTE: Any number of IF…THEN…ELSE…END_IF commands may be nested to generate complex selection commands.

Example IF...THEN...END_IF

If FLAG is 1, the instructions will be executed; if FLAG is 0, they will not be executed.

```
IF FLAG THEN
   C:=SIN_REAL(A) * COS_REAL(B) ;
   B:=C - A ;
END_IF ;
```

Example IF NOT...THEN...END_IF

Using NOT, the condition may be inverted (execution of both instructions at 0).

```
IF NOT FLAG THEN
   C:=SIN_REAL(A) * COS_REAL(B) ;
   B:=C - A ;
END_IF ;
```

Related topic(s)

ELSE *(see page 429)*

ELSEIF *(see page 430)*

ELSE

Description

The ELSE command always comes after an IF…THEN-, ELSIF…THEN- or CASE-command.

If the ELSE command comes after IF or ELSIF, the command or group of commands will only be executed if the associated Boolean expressions of the IF and ELSIF command have the 0 value (false). If the condition of the IF or ELSIF command is 1 (true), the command or group of commands will not be executed.

If the ELSE command comes after CASE, the command or group of commands will only be executed if no identification contains the value of the selector. If an identification contains the value of the selector, the command or group of commands will not be executed.

NOTE: As many IF…THEN…ELSE…END_IF-commands as required can be encapsulated to create complex selection commands.

Example ELSE

```
IF A>B THEN
   C:=SIN_REAL(A) * COS_REAL(B) ;
   B:=C - A ;
ELSE
   C:=A + B ;
   B:=C - A ;
END_IF ;
```

Related topic(s)

IF *(see page 428)*

ELSIF *(see page 430)*

CASE *(see page 431)*

ELSIF...THEN

Description

The ELSIF-command always comes after an IF…THEN-command. The ELSIF-command establishes that a command or group of commands will only be executed if the associated Boolean expression of the IF-command has the 0 value (false) and the associated Boolean expression of the ELSIF command has the 1 value (true). If the condition of the IF-command is 1 (true) or the condition of the ELSIF-command is 0 (false), the command or group of commands will not be executed.

The THEN-command identifies the end of the ELSIF-condition(s) and the beginning of the command(s).

NOTE: As many IF…THEN…ELSIF…THEN…END_IF-commands as required can be encapsulated to create complex selection commands.

Example ELSIF…THEN

```
IF A>B THEN
   C:=SIN_REAL(A) * COS_REAL(B) ;
   B:=SUB_REAL(C,A) ;
ELSIF A=B THEN
   C:=ADD_REAL(A,B) ;
   B:=MUL_REAL(C,A) ;
END_IF ;
```

Example encapsulated commands

```
IF A>B THEN
   IF B=C THEN
    C:=SIN_REAL(A) * COS_REAL(B) ;
   ELSE
    B:=SUB_REAL(C,A) ;
   END_IF ;
ELSIF A=B THEN
   C:=ADD_REAL(A,B) ;
   B:=MUL_REAL(C,A) ;
ELSE
   C:= DIV_REAL (A,B) ;
END_IF ;
```

Related topic(s)

IF *(see page 428)*

ELSE *(see page 429)*

CASE...OF...END_CASE

Description

The CASE instruction consists of an INT data type expression (the "selector") and a list of instruction groups. Each group is provided with a marke which consists of one or several whole numbers (ANY_INT) or zones of whole number values. The first group is executed by instructions, whose marke contains the calculated value of the selector. Otherwise none of the instructions will be executed.

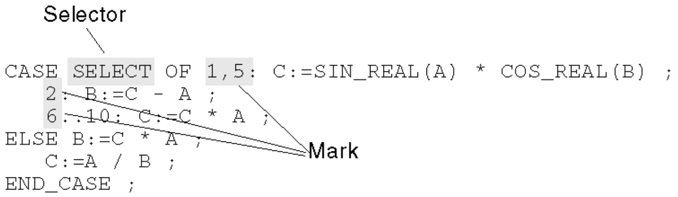
The OF instruction indicates the start of the mark.

An ELSE instruction may be carried out within the CASE instruction, whose instructions are executed if no mark contains the selector value.

The END_CASE instruction marks the end of the instruction(s).

Example CASE...OF...END_CASE

Example CASE...OF...END_CASE



```
CASE SELECT OF 1,5: C:=SIN_REAL(A) * COS_REAL(B) ;
   2: B:=C - A ;
   6..10: C:=C * A ;
ELSE B:=C * A ;
   C:=A / B ;
END_CASE ;
```

Related topic(s)

ELSE *(see page 429)*

FOR...TO...BY...DO...END_FOR

Description

The FOR instruction is used when the number of occurrences can be determined in advance. Otherwise WHILE *(see page 435)* or REPEAT *(see page 437)* are used

The FOR instruction repeats an instruction sequence until the END_FOR instruction. The number of occurrences is determined by start value, end value and control variable. Start value, end value and the control variable must be the same type of data (DINT or INT) and may not be modified by one of the repeated instructions. The FOR instruction increments the control variable value of one start value to an end value. The increment value has the default value 1. If a different value is to be used, it is possible to specify an explicit increment value (variable or constant). The control variable value is checked before each renewed loop running. If it is outside the start value and end value range, the loop will be left.

Before running the loop for the first time a check is made to determine whether incrementation of the control variables, starting from the initial value, is moving towards the end value. If this is not the case (e.g. initial value ≤end value and negative increment), the loop will not be processed.

Using this ruler, continuous loops will be prevented.

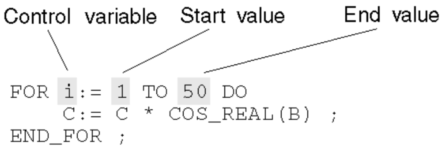
NOTE: For the end value of the data type DINT the range of values -2 147 483 646 to 2 147 483 645 will apply.

The DO command identifies the end of the repeat definition and the beginning of the instruction(s).

Repetition may be terminated early by using the EXIT instruction. The END_FOR instruction marks the end of the instruction(s).

Example: FOR with increment "1"

FOR with increment "1"



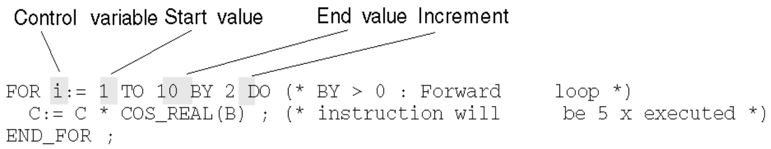
```
FOR i:= 1 TO 50 DO
    C:= C * COS_REAL(B) ;
END_FOR ;
```

FOR with increment not equal to "1"

If an increment other than "1" is to be used, this can be defined by BY. The increment, the initial value, the end value, and the control variable must be of the same data type (DINT or INT). The criterion for the processing direction (forward, backward) is the sign of the BY expression. If this expression is positive, the loop will run forward; if it is negative, the loop will run backward.

Example: Counting forward in two steps

Counting forward in two steps



```
FOR i:= 1 TO 10 BY 2 DO (* BY > 0 : Forward     loop *)
  C:= C * COS_REAL(B) ; (* instruction will     be 5 x executed *)
END_FOR ;
```

Example: Counting backward

Counting backward

```
FOR i:= 10 TO 1 BY -1 DO (* BY < 0 : Backward loop *)
  C:= C * COS_REAL(B) ; (* Application will be executed 10
x *)
END_FOR ;
```

Example: "Unique" loops

The loops in the example are run once precisely, as the initial value = end value. In this context it does not matter whether the increment is positive or negarive.

```
FOR i:= 10 TO 10 DO (* Unique Loop *)
  C:= C * COS_REAL(B) ;
END_FOR ;
```

or

```
FOR i:= 10 TO 10 BY -1 DO (* Unique Loop *)
  C:= C * COS_REAL(B) ;
END_FOR ;
```

Example: Critical loops

If in the example there is the increment j > 0, the instructions will not be executed, as the situation initial value > end value only permits an increment ≤0. A continuous loop can only arise if the increment is 0. If this situation is identified during the section analysis, an error message will be generated. If the error is identified during running time, an error message will be generated in the event viewer.

```
FOR i:= 10 TO 1 BY j DO (* Backward loop *)
  C:= C * COS_REAL(B) ;
END_FOR ;
```

If in the example there is the increment j < 0, the instructions will not be executed, as the situation initial value < end value only permits an increment ≥ 0. A continuous loop can only arise if the increment is 0. If this situation is identified during the section analysis, an error message will be generated. If the error is identified during running time, an error message will be generated in the event viewer.

```
FOR i:= 1 TO 10 BY j DO (* Forward loop *)
  C:= C * COS_REAL(B) ;
END_FOR ;
```

Example: Illegal loops

Illegal loops

```
FOR i:= 1 TO 10 BY 0 DO (* Error with Section- *)
  C:= C * COS_REAL(B) ; (* Analysis, as continous loop *)
END_FOR ;
```

or

```
FOR i:= 1 TO 10 BY j DO (* at j=0, Error message *)
  C:= C * COS_REAL(B) ; (* in of Event indicator *)
END_FOR ;
```

WHILE...DO...END_WHILE

Description

The WHILE instruction has the effect that a sequence of instructions will be executed repeatedly until its related Boolean expression is 0 (false). If the expression is false right from the start, the group of instructions will not be executed at all.

The DO command identifies the end of the repeat definition and the beginning of the command(s).

The occurrence may be terminated early using the EXIT.

The END_WHILE instruction marks the end of the instruction(s).

⚠ WARNING

Risk of program crashing

WHILE must not be used to carry out synchronization between processes, e.g. as a "waiting loop" with an externally determined end condition. This means that a continous loop must not be created, unless you prevent this using the function **Project →Code generation options... →Enable Loop Control**.

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

⚠ WARNING

Risk of program crashing

WHILE must not be used in an algorithm for which fullfilling the loop end condition or the execution of an EXIT instruction can not be guaranteed. This means that a continuous loop must not be created, as this may result in crashing the program, unless you prevent this by using the function **Project →Code generation options... →Enable Loop Control**.

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

Example WHILE...DO...END_WHILE

```
    var := 1
WHILE var <= 100 DO
   var := var + 4;
END_WHILE ;
```

Related topic(s)

EXIT *(see page 438)*

REPEAT...UNTIL...END_REPEAT

Description

The REPEAT instruction has the effect that a sequence of instructions is executed repeatedly (at least once), until its related Boolean condition is 1 (true).

The UNTIL instruction marks the end condition.

The occurrence may be terminated early using the EXIT.

The END_REPEAT instruction marks the end of the instruction(s).

⚠ WARNING

Risk of program crashing

REPEAT must not be used to carry out synchronization between processes, e.g. as a "waiting loop" with an externally determined end condition. This means that a continous loop must not be created, unless you prevent this using the function **Project →Code generation options... →Enable Loop Control**.

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

⚠ WARNING

Risk of program crashing

REPEAT must not be used in an algorithm for which fullfilling the loop end condition or the execution of an EXIT instruction can not be guaranteed. This means that a continuous loop must not be created, as this may result in crashing the program, unless you prevent this by using the function **Project →Code generation options... →Enable Loop Control**.

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

Example REPEAT...UNTIL...END_REPEAT

```
    var := -1
REPEAT
   var := var + 2
   UNTIL var >= 101
END_REPEAT ;
```

Related topic(s)

EXIT *(see page 438)*

EXIT

Description

The EXIT command is used to terminate repeat instructions (FOR, WHILE, REPEAT) before the end condition has been met.

If the EXIT instruction is within a nested occurrence, the innermost loop (in which EXIT is situated) is left. Next, the first instruction following the loop end (END_FOR, END_WHILE or END_REPEAT) is executed.

Example EXIT

If FLAG has the value 0, SUM will be 15 following execution of the instructions.

If FLAG has the value 1, SUM will be 6 following execution of the instructions.

```
SUM : = 0 ;
FOR I := 1 TO 3 DO
   FOR I := 1 TO 2 DO
      IF FLAG=1 THEN EXIT;
      END_IF;
      SUM := SUM + J ;
   END_FOR ;
   SUM := SUM + I ;
END_FOR
```

Related topic(s)

CASE *(see page 431)*

WHILE *(see page 435)*

REPEAT *(see page 437)*

Empty instruction

Description

Empty instructions are generated by a semicolon (;).

Comment

Description

Within the ST editor, comments start with the string (* and end in the string *). Any comments may be entered between these two strings. Comments may be entered in any position in the ST editor. Comments are shown in colour.

NOTE: In accordance with IEC 1131-1, nested comments are not permissible. However, if you wish to place theses elsewhere, you can release them by using **Options →Preferences →IEC Extensions →Allow nested comments**.

11.5 Call up of functions, Function Blocks (EFBs) and Derived Function Blocks (DFBs)

Overview

This section describes the call up of functions, Function Blocks (EFBs) and Derived Function Blocks (DFBs).

What's in this Section?

This section contains the following topics:

Topic	Page
Function Block/DFB Invocation	442
Function Invocation	446

Function Block/DFB Invocation

Use of Function Blocks and DFBs

Function blocks are provided by Concept in the form of libraries. The logic of the function blocks is created in C++ programming language and cannot be altered in the ST Editor. The names of the available function blocks can be taken from the block libraries.

DFBs are function blocks which can be defined in Concept-DFB. There is no difference between functions and function blocks for DFBs. They are always handled as function blocks regardless of their internal structure.

The use of function blocks and DFBs consists of three parts in ST:
- the declaration *(see page 443)*,
- the function block/DFB invocation *(see page 443)*,
- the use of the function block/DFB outputs *(see page 445)*.

NOTE: The declaration of the function block/DFB invocation can take place manually or you can create the block end and the assignment of the parameters using the menu command **Objects** → **Insert FFB**.

Function Blocks with Limited Use

Use of the following EFBs from the DIAGNO block library is limited in ST (function blocks can be used, but the expanded diagnostic information cannot be evaluated):
- XACT, XACT_DIA
- XDYN_DIA
- XGRP_DIA
- XLOCK,
- XPRE_DIA
- XLOCK_DIA
- XREA_DIA

Function Blocks with Limited Invocation

For EFBs which have one or more outputs with data type ANY but no inputs with data type ANY (generic outputs/inputs), the block invocation can only take place in compact form *(see page 445)*. e.g. in the block library **LIB984**:
- GET_3X
- GET_4X

Unusable Function Blocks

Unusable Function Blocks:

- EFBs which use several registers with only the entry for the first register on the input/output (e.g. MBP_MSTR from the COMM block library) cannot be used.
- EFBs which contain outputs with input information (e.g. GET_BIT, R2T from the LIB984 block library) cannot be used
- The following EFBs from the **COMM** block library cannot be used for the technical reasons listed above:
  - CREADREG
  - CREAD_REG
  - CWRITREG
  - CWRITE_REG
  - READREG
  - READ_REG
  - WRITEREG
  - WRITE_REG
  - MBP_MSTR
- The following EFBs from the **LIB984** block library cannot be used for the technical reasons listed above:
  - FIFO
  - GET_BIT
  - IEC_BMDI
  - LIFO
  - R2T
  - SET_BIT
  - SRCH
  - T2T

Declaration

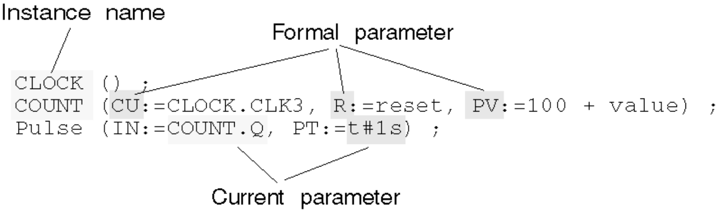
Before invoking the function block/DFBs, they must be declared using VAR and END_VAR *(see page 426)*.

Function Block/DFB Invocation

Function blocks/DFBs are invoked using an instruction consisting of the instance name for the FB/DFB, which is followed by a list, in brackets, of value assignments (current parameters) to formal parameters. The order of the formal parameters in a function block invocation is not significant. It is not necessary for all formal parameters to be assigned a value. If a formal parameter is not assigned a value, the initial value defined in the variable editor is used when executing the function block. If an initial value is not defined, the default value (0) is used.

NOTE: Inputs of type VARINOUT *(see page 491)* always have to be assigned a value.

Function block/DFB invocation:



```
CLOCK () ;
COUNT (CU:=CLOCK.CLK3, R:=reset, PV:=100 + value) ;
Pulse (IN:=COUNT.Q, PT:=t#1s) ;
```

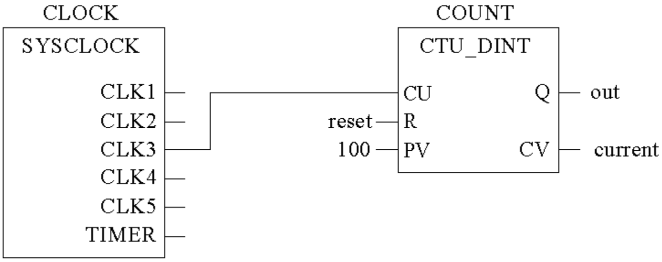
NOTE: In ST, unlike the graphic programming languages (FBD, LD), FB/DFB instances can be called multiple times.

NOTE: Even if the function block has no inputs or the input parameters are not to be defined, the function block should be invoked before the outputs can be used. Otherwise the initial values for the outputs are given, i.e. "0".

Declaration and invocation of a function block in ST:

```
VAR
   CLOCK : SYSCLOCK ;
   COUNT : CTU_DINT ;
END_VAR

   CLOCK () ;
   COUNT (CU:=CLOCK.CLK3, R:=reset, PV:=100) ;
   out:=COUNT.Q ;
   current:=COUNT.CV ;
```

Invocation of the function block in FBD.



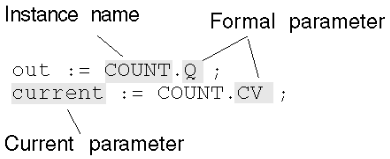
Function Block/DFB Invocation in Compact Form

The block invocation and the assignments for the inputs/outputs are also possible in a more compact form, which saves runtime:

```
VAR
   CLOCK : SYSCLOCK ;
   COUNT : CTU_DINT ;
END_VAR

   CLOCK () ;
   COUNT (CU:=CLOCK.CLK3, R:=reset, PV:=100,
          Q=>out, CV=>current) ;
```

Use of the Function Block/DFB Outputs

The outputs of the function block/DFBs can always be used when a variable (read only) can also be used.



```
out := COUNT.Q ;
current := COUNT.CV ;
```

Function Invocation

Using Functions

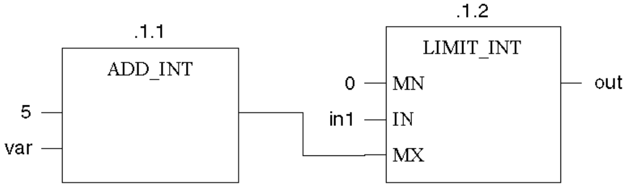
Functions are provided by Concept in the form of libraries. The logic of the function is created in C++ and cannot be edited in the ST Editor. The names of the available function can be taken from the block libraries.

NOTE: The declaration of the function invocation can take place manually or you can create the block end and the assignment of the parameters using the menu command **Objects** → **Insert FFB**.

Invoking a function in ST:

```
out := LIMIT_INT (MN:=0, IN:=in1, MX:=5 + var) ;
```

Invoking the function FBD:

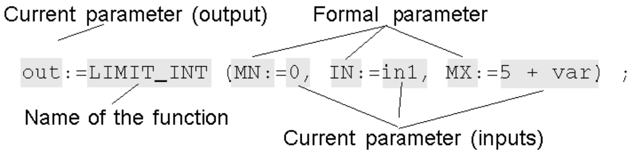


Unusable Functions

Functions which have one or more outputs with data type ANY but no inputs with data type ANY (generic outputs/inputs) cannot be used in ST.

Invoking a Function: Variant 1

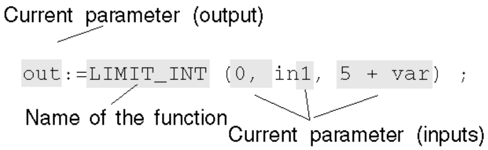
The function can also be invoked using an instruction consisting of a current parameter (variable) followed by the instruction assignment ":=" followed by the name of the function followed by a list of value assignments (current parameters) for the formal parameters in brackets. The order of the formal parameters in a function block invocation is not significant.



```
out:=LIMIT_INT (MN:=0, IN:=in1, MX:=5 + var) ;
```

Invoking a Function: Variant 2

Functions are invoked using an instruction. The instruction consists of the current parameter (variable) for the output followed by the instruction assignment ":=" followed by the name of the function followed by a list of current input parameters in brackets. The order of the current parameters in a function invocation is significant.



```
out:=LIMIT_INT (0, in1, 5 + var) ;
```

11.6 Syntax check and code generation

Overview

This section describes the syntax check and the code generation of the structured ST text.

What's in this Section?

This section contains the following topics:

Topic	Page
Syntax Check	449
Code generation	450

Syntax Check

Introduction

A syntax check can be performed during the program/DFB creation with **Project → Analyze section**.

Syntax Check Options

With the menu command **Options →Preferences →IEC extensions... →IEC-extensions** the syntax check options can be defined.

NOTE: The settings in this dialog are used in the project description (PRJ.DSK) and in the Concept installations description (CONCEPT.DSK), i.e. they are valid for the entire Concept installation.

If a project is opened, that was created using different settings (e.g. **Allow nested comments** in the project and not in the actual Concept Installation), errors can occur when opening the project.

Allow Case Insensitive Keywords

If the check box **Allow case insensitive keywords** is checked, upper and lower case for all keywords is enabled.

Allow nested comments

If the check box **Nested comments authorized** is checked, nested comments can be entered. There are no limits to the nesting depths.

Allow Leading Digits in Identifiers

If the check box **Allow leading digits in identifiers** is checked, figures as the first character of identifiers (i.e. variable names, step names, EFB names) are possible. Identifiers, which consist solely of figures are, however, not authorized, they must contain at least one letter.

Unused Parameters Cause Warnings

The IEC 1131-3 permits functions and Function Blocks to be called up without calling up the assignment of all the input parameters. These unused parameters are implicitly assigned a 0, or they retain the value from the last call up (Function Blocks only).

If in the menu command **Options →Preferences →Analysis... →Analysis** the check box **Unused parameters lead to warnings** is checked, a list of these unused parameters is displayed in the message window when generating the code.

Code generation

At a Glance

The menu command **Project →Options for code generation** is used to define options for code generation.

Fastest code (restricted check)

If the check box **Fastest Code (restricted check)** is activated, a runtime-optimized code will be generated.

This runtime optimization is achieved with integer arithmetic (e.g. "+" or "-") with simple process commands instead of EFB calls.

Process commands are much faster than EFB calls, but they generate no error messages, such as e.g. Arithmetic or Array overrun. This option should only be used if it has been ensured that the program is free of arithmetic errors.

Example: Fastest code

```
IF i <= max THEN    (*i and max are of INT type*)
   i := i +1 ;
END_IF;
```

If **Fastest Code (restricted check)** is selected, the addition "i1 + 1" is executed with the process command "add". The code is now faster than if EFB ADD_INT had been called up. However, no runtime error is generated if "max" is 32767. In this case, "i" would overrun from 32767 to -32768!

Activate loop control

This check box activates a software watchdog for continuous loops.

If this check box is checked, with loops within IL and ST sections, it is tested whether these loops are again exited within a certain time. The time authorized depends on the manually defined watchdog time. The authorized time for **all loops** combined constitutes 80% of the Hardware watchdog time. In this way triggering of the hardware watchdog by endless loops is disabled. If a time consuming loop or an endless loop is detected, processing of the affected section will stop, an entry in the Event display will be generated and processing of the next section will begin. In the next cycle, the segment will be re-processed until a time consuming loop or an endless loop is detected once again, or until the segment is finished correctly.

NOTE: If the hardware watchdog stops the PLC when a time consuming loop or an endless loop is detected, this option should not be activated. The hardware watchdog itself is not switched off by this function.

11.7 Online functions of the ST programming language

Online functions

Description

The online functions available in the programming language Instruction List (IL) are available here (see *Online functions of the IL instruction list, page 385*).

11.8 Creating a program with the structured ST text

Creating a program in structured ST text

At a Glance

The following description contains an example of the creation of a program in the programming language of structured ST text. This creation is divided into 2 main steps:

Step	Action
1	Generating a section *(see page 452)*
2	Creating the logic *(see page 453)*

Generating a section

The procedure for generating a section is as follows:

Step	Action
1	Using the menu command **File** →**New section...** and enter a section name. **Note:** The section name (max. 32 characters) is not case-sensitive and must be unique throughout the project. If the name entered already exists, a warning is given and another name must be chosen. The section name must correspond to the IEC name conventions, otherwise an error message is displayed. **Note:** In accordance with IEC1131-3, only letters are permitted as the first character of names. Should numbers be required as the first character, however, the menu command **Options** →**Presettings** →**IEC expansions...** →**Enable leading figures in identifiers** .

Creating the logic

The procedure for creating the logic is as follows:

Step	Action

1	Declare the Function Block and DFBs, which are to be used, with assistance from VAR…END_VAR.
	Example:
	`VAR`
	`    RAMP_UP, RAMP_DOWN, RAMP_X : TON`
	`    COUNT : CTU_DINT ;`
	`END_VAR`

2	Declare the variables and their initial value in the Variable Editor.

3	Create the logic of the program.
	Example:
	`SUM : = 0 ;`
	`FOR I = 1 TO 3 DO`
	`   FOR J = 1 TO 2 DO`
	`      IF FLAG = 1 THEN EXIT;`
	`      END_IF;`
	`      SUM := SUM + J ;`
	`   END_FOR ;`
	`   SUM = SUM + I ;`
	`END_FOR`

4	Save the section with the menu command **Data file** →**Save project** .

Ladder Logic 984

12

Introduction

This chapter describes the programming language Ladder Logic 984.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
12.1	General about Ladder Logic 984	456
12.2	Working with Ladder Logic 984	458
12.3	Subroutines	469
12.4	Equation Network Editor	471
12.5	LL984 Programming Modes	480

12.1 General about Ladder Logic 984

General about Ladder Logic 984

Introduction

Ladder logic is displayed in a graphic window. Each window contains exactly one ladder logic section. One or more different ladder sections can be viewed or edited (multiple windows of the same section is not supported).

If you are adding a new section the section number is posted for your reference.

Correlation between Sections and Segments

Each ladder logic section becomes tied to a PLC ladder logic segment (e.g., one section equals one segment) by a segment number entry in the **Section Properties** dialog.

One network at a time is visible in each section.

Using the Keyboard

Editing in Concept is ordinarily done with the mouse, but it is also possible with the keyboard (see also *Short Cut Keys in the LL984-Editor, page 847*).

Project Analyzation

Ladder logic is analyzed before the program is downloaded to the controller.

The editor permits only valid Ladder Logic to be entered in the editor, e.g.:

- Only those logic elements supported by the current PLC configuration are visible for selection. You must configure the controller before entering logic.
- The analyzer does not allow references outside the range of the current configuration.
- The analyzer does not allow duplicate coils unless supported by the current configuration.
- The analyzer does not allow loadables that are not in the current configuration.
- All subroutines must exist in a single section.
- The section containing subroutines cannot be scheduled.
- All jumptosubroutine instructions must reference the same section.
- Multiple variables per reference are supported. A user preference is available to enable or disable this feature. When multiple variables are declared for a given reference either a warning or error is generated, depending on this preference.

NOTE: Your changes to configuration may cause the program to become incompatible with the configuration.

NOTE: Contacts or coils may be entered without references. This not allowed, but not covered by the project analyzation.

Capacity and Limitations

Capacity and Limitations:

- Editor cannot permit more sections than number of segments
- Editor cannot permit more networks than can fit in controller memory

12.2 Working with Ladder Logic 984

Introduction

This section describes how to work with Ladder Logic 984.

What's in this Section?

This section contains the following topics:

Topic	Page
Entering and Editing Logic Objects	459
Entering and Editing Variables	461
Ladder and Network Editing	464
Reference Zoom and DX Zoom	466
Search and Replace	468

Entering and Editing Logic Objects

Prerequisite Requirements

Only those logic objects supported by the current PLC configuration are visible for your selection. You must configure the controller before entering logic.

For Loadables that require settings in **Project** →**Configurator** →**Configure** → **Config. extensions**, provisions must be made before inclusion in a Ladder program.

Navigation

When you are in the middle of a section, the next or previous network can be viewed by scrolling with **PgUp** and **PgDn** keys.

When you are at the top or bottom of a section, the next or previous section can be viewed by scrolling with **PgUp** and **PgDn** keys, if the section exists.

For instance if you are at the end of networks in the last section (and it is not section 32), you are prompted with a dialog to allow appending a new section. Each network is compared against the database on **PgUp**/**PgDn** (in Combo-Mode).

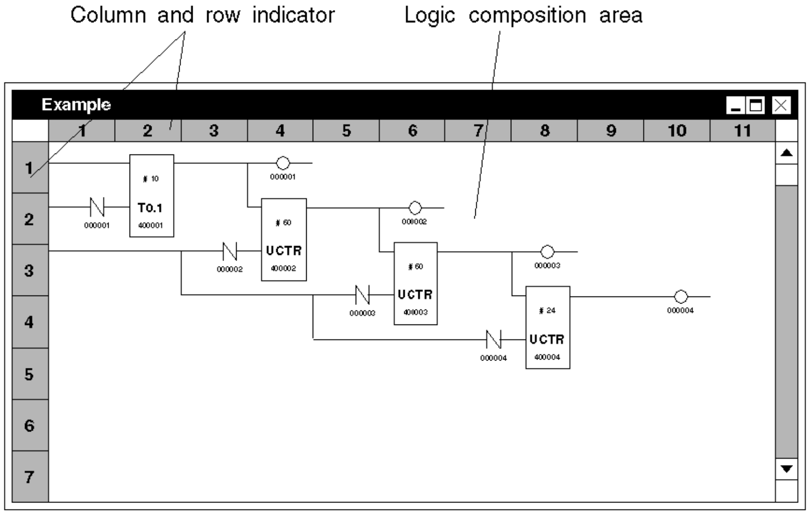
You can go to a network within a section by using the **Go to Network** dialog. You can select the first or last network within the current section, or go to a network by entering a network name or number. A sortable list of networks (with names) is provided.

Dialog Interaction

Your actions for entering and editing Ladder Logic follow the standards of MS-Windows and conventions of major MS-Windows applications. When an element is selected with the mouse, the mouse cursor changes to a graphical picture that represents the logic item. The application programmer places the logic item in the edit area by clicking or pressing the **Enter** key.

A keyboard cursor is shown as a high lighted cell (block) within the Ladder Logic network. For each editing mouse action there is a corresponding keyboard action (see also *Short Cut Keys in the LL984-Editor, page 847*). When the keyboard is used to enter a logic item, there is no initial selection step the logic item is immediately placed in the network at the keyboard cursor.

Ladder Logic sample network:



Placing Objects

The entire range of programming objects is available from the **Object** main menu and selected sub menu items.

Occupied nodes of equivalent height can be overwritten.

Instructions can be entered by typing the name in a dialog.

NOTE: When possible, Concept uses **Ctrl** key in place of the Modsoft **Alt** key (see also *Modsoft Keys with Concept Equivalents, page 1008*).

Online Restriction

Online restrictions:

- Online deletes require user confirmation.
- Concept does not support drag/drop of programmed elements when online.

Entering and Editing Variables

Introduction

References of nodes in logic items can be viewed or edited by double clicking an item in a network or by pressing the **Enter** key on an item that has the focus. An **Object Properties** dialog is presented when you double click on a highlighted object or by pressing the **Enter** key on an item that has the focus.

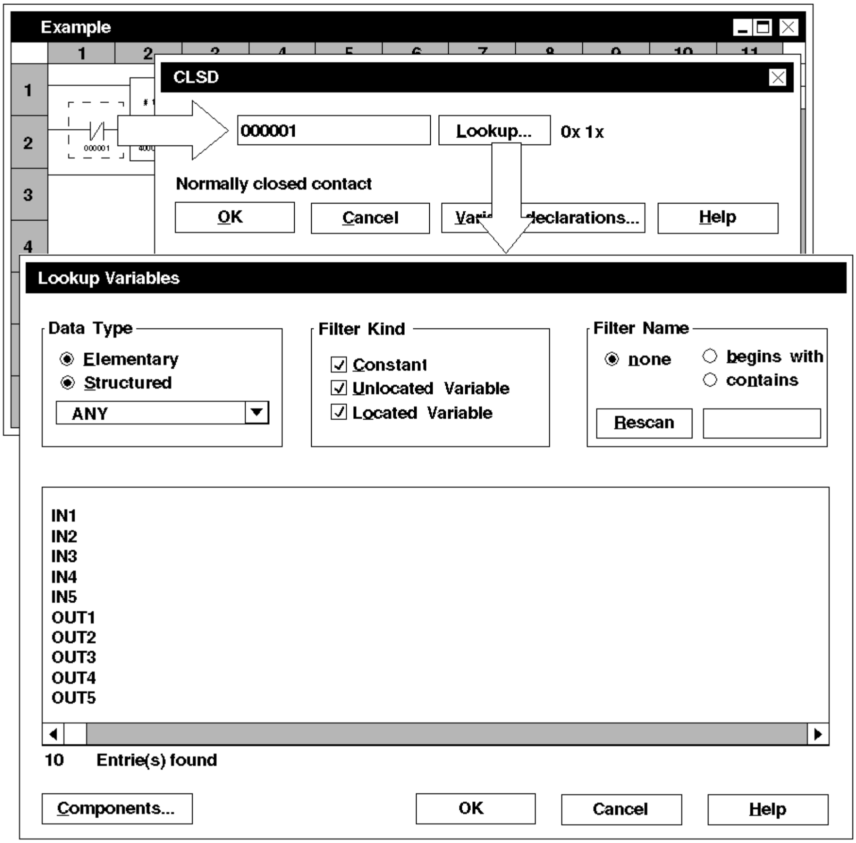
You can view the already created variables by clicking on the **Lookup** button.

You can create new variables by clicking on the **Variable declarations** button.

Editing References

References of each node of the logic element (e.g., multi-node) can be edited. When applicable, you can enter the sub-function name (from a drop-down list). If both a constant and a reference can be entered, the # sign must be entered before a constant beginning with 0, 1, 3, or 4. You may enter a variable name for references.

Object properties with **Lookup Variables** dialog:



000001 0x 1x

Normally closed contact

Data Type

● Elementary

● Structured

ANY

Filter Kind

☑ Constant

☑ Unlocated Variable

☑ Located Variable

Filter Name

● none ○ begins with ○ contains

IN1
IN2
IN3
IN4
IN5
OUT1
OUT2
OUT3
OUT4
OUT5

10 Entrie(s) found

Entry Format of Reference Values

When entering references, the first digit is always the reference type (e.g., 0x) and the following digits are the reference number. You may change the format of the displayed references by setting **Options** →**Preferences** →**Common**.

Status Bar

The variable name (if applicable) is shown on the display status line, for the element in focus. When online, the value of the reference is also shown. The initial display format of the reference value depends on the instruction in the program. You can change the display format using the following keys in combination to define the data precision and then format.

Table of display formats:

Precision	Format
L (32bit)	D (signed decimal)
	U (unsigned)
	A (ascii)
	H (hex)
S (16bit)	D (signed decimal)
	U (unsigned)
	A (ascii)
	H (hex)

Reference Offsetting

Program references can be offset using **Edit →Offset References**. Multiple references can be offset in the same step (while offline). Sections/networks that are being offset are selectable. You are asked to put in the first and last reference to be affected and put in the number you want the offset to be.

Ladder and Network Editing

Introduction

Ladder and network edit functions are available from the main menus **Edit** and **Networks**.

NOTE: Menu items in diminished brightness are not selectable given the current configuration, status, etc,.

Undo Delete

The **Edit** →**Undo delete** function, is an ofline mode function that allows up to the most current 5 deletes to be undone. The **Undo delete** is provided for each ladder logic section and includes element and network cut/delete events.

Insert, **Append** or **Reorder** network operations cause a reset of the delete-save area therby assuring the network numbers are not contaminated.

Select/De-select All, Cut, Copy and Paste

Select all, **Cut**, **Copy**, and **Paste** operations on individual language elements occur within a single network (at a time). Your can select-all or unselect-all elements in a single network. You can also select, cut, copy, and paste language elements within and between ladder logic networks or sections.

In an online paste operation, the item being pasted is done in increments of scans until complete.

Selecting Elements

You cannot select multiple language elements (e.g., accumulate selections) across networks or sections.

Setting focus to an element is done by moving the cursor (either with mouse or arrow key) to the element.

Selection of elements is done by clicking or pressing the **Spacebar** key on the element which has the focus.

Multiple elements can be selected by using mouse-rubber-band actions. Multiple elements can also be selected by holding down the **Shift** key and then clicking on the elements or pressing the **Spacebar** key on the elements.

An entire row or column can be selected by clicking on the rung or column header in the network.

The mouse provides a finer level of selection than the keyboard. If two or more elements appear in a cell (e.g., both a vertical short and a contact), pressing the **Spacebar** key selects all items in that cell. Clicking the mouse selects the element closest to the mouse pointer.

Open Row

A new row is opened at the current cursor position. This command is executed only if there is enough free space (i.e., the last row is empty). The rest of the network is shifted down accordingly. Function boxes and other objects with a height of more than one node are not split by this command.

Open Column

If the rightmost node column is free, the rest of the network is shifted right, and an empty column is opened at the current cursor position.

Close Row

If the node row on which the cursor is positioned is empty, all node elements below are shifted up one row, and an empty bottom row remains.

Close Column

If the node column on which the cursor is positioned is empty, all node elements to the right are shifted left one column and an empty right column remains.

Network

By using the **Networks** main menu and it’s subcommands, you can insert (before) or append (after) a single empty network or delete one or more networks.

In addition, within a single section, you can cut/copy a network then you can copy/paste networks in any section. You are provided with a list of networks to consider for the cut/copy operation

Reorder Networks

Network execution reordering is an offline function. You may change the execution order of networks within a single section. Networks are solved in the order they appear in the section.

The execution order of networks is changed by using the **Network Execution Order** dialog. i.e. select **Network** →**Reorder...**.

Network Comments

A section description can be included. Each network can be individually commented using network comments and online comments.

A network name can be entered in the **Network Comment** dialog.

Reference Zoom and DX Zoom

Introduction

Concept offers you two different zoom types:
- the Reference Zoom
- the DX Zoom

Reference Zoom

Some programming elements allow parameters to be set which in effect customize a network implementation for this specific element. Such features as ranges and limits etc., are input using this zoom edit capability.

Information on individual references can be viewed or edited.

The **Reference Zoom** dialog shows the following information about a reference:
- State-ram value
- The drop/rack/slot if the reference is in I/O map
- If reference is 0x or 1x, then the disable/enable state is shown

The initial display format of 3x and 4x reference values depends on the instruction in the program. The display format can be changed. The state ram value or disable/enable state (if applicable) can also be changed. Constants cannot be zoomed. You cannot zoom on variables without a reference. Reference Zoom dialogs can be used for 4x references and for 0x references that are disabled.

DX Zoom

The DX Zoom editor allows you to edit registers for DX functions. These registers used by the DX function also have text descriptions associated with them to aid with DX programming. There is both keyboard and mouse access to DX zoom from the Ladder Logic editor.

The **DX Zoom** dialog allows you to edit registers for given DX functions. The DX zoom screen contains text for each register, bit, or group of bits.

The allowed data types are:

Data Type	**Length**
Unsigned Integer	16 bit
Signed Integer	16 bit
Unsigned Long Integer	32 bit
Signed Long Integer	32 bit
float	32 bit
bit (flag)	1 bit
bitfield	1-16 bits

The allowed complex data types are:

Complex Data Types	Length
equation	1-16 bits
ASCII	String up to 80 characters

Absolute addressing is the only addressing method allowed. There is no support for indirect addressing

In addition to data entry, DX zoom has the capability to display textual information associated with a particular register. Each register entry will have an associated descriptor as well as context sensitive help.

Search and Replace

Trace

The **Online →Trace** function finds coils from 0x references in the program. You can trace a coil by first setting focus to a 0x reference and then running the trace function. The result of trace is to position the network with the found coil on the edit area. After a successful trace, with **Online →ReTrace** you can go back to the initial 0x reference.

Online Search

A separate dialog is available for **Project →Search** in direct mode. The **Online Search** dialog. On each find, the choice to search previous or next is provided. Search can be canceled at any time.

There is no support for searching variable names if in Ladder Logic direct mode.

Replace References

Search and replace of references occur throughout an entire program. You can select which sections/networks are being searched.

The **Edit →Replace References** dialog is modal. Request may be prompted for each individual replace, or request to replace all with no prompting. Replaced references are listed in the **Project →Search →Search History** list.

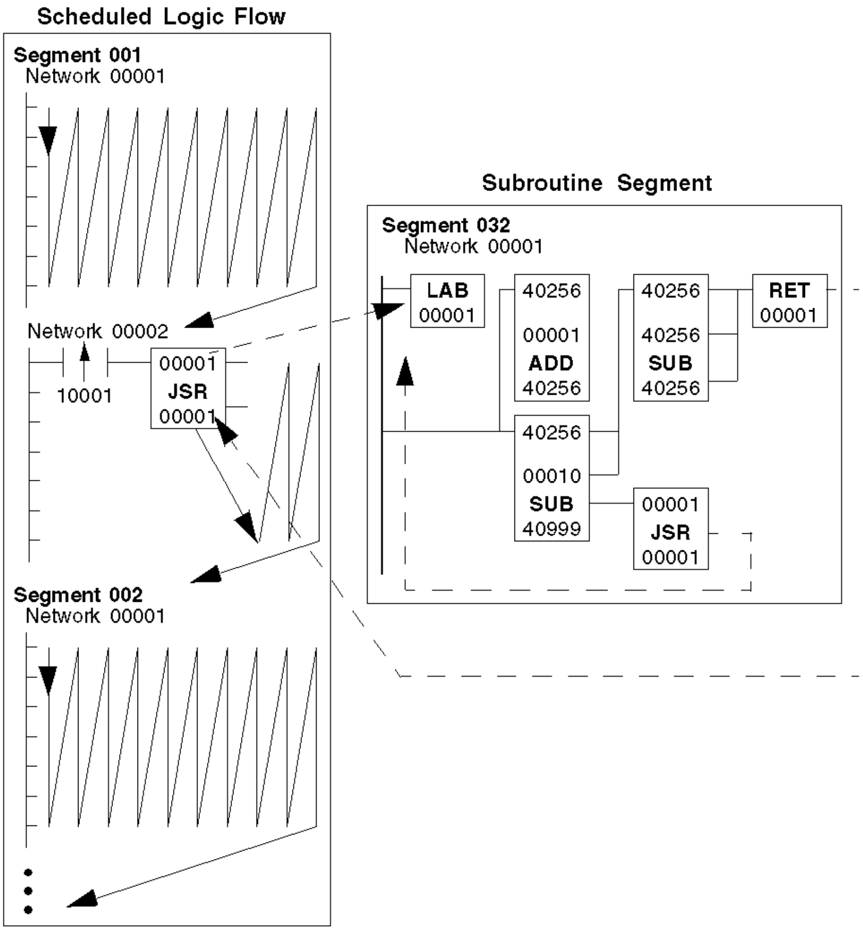
You may exclude DX functions with discrete references from the search. DX functions require 0x and 1x references to be on a 16 bit boundary.

12.3 Subroutines

Subroutines

Example

The example below shows a series of three user logic networks, the last of which is used for an up-counting subroutine. Segment 32 has been removed from the order-of-solve table in the segment scheduler.



Description of Example

Description of example:

Stage	Description
1	When input 10001 to the JSR block in network 2 of segment 1 transitions from OFF to ON, the logic scan jumps to subroutine #1 in network 1 of segment 32. **Result:** The subroutine will internally loop on itself ten times, counted by the ADD block.
2	The first nine loops end with the JSR block in the subroutine (network 1 of segment 32) sending the scan back to the LAB block.
3	Upon completion of the tenth loop, the RET block sends the logic scan back to the scheduled logic at the JSR node in network 2 of segment 1.

12.4 Equation Network Editor

Introduction

This section describes the LL984 equation network editor.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction	472
Equation Editing	474
Syntax and Semantics	476

Introduction

Overview

The equation network is a combination of both Ladder Logic and an algebraic equation. This network type allows a control designer to incorporate an algebraic equation into a Ladder Logic program The **Equation Network Editor** dialog has no row/column numbers since they have no significance. The grid display option is not available for the equation network because the row/column concept does not apply to this new network type. You have the ability, using Ladder Logic notation, to indicate when the equation will be solved.

Equation network is a special type of Ladder Logic network that allows you to specify the value of a result register in algebraic notation. If your PLC has an floating point processor, equation network will take advantage of this feature for faster processing. It uses a full Ladder Logic network to compose the equation, with a contact or horizontal short as the enabling input and up to five output coils to describe the state of the result.

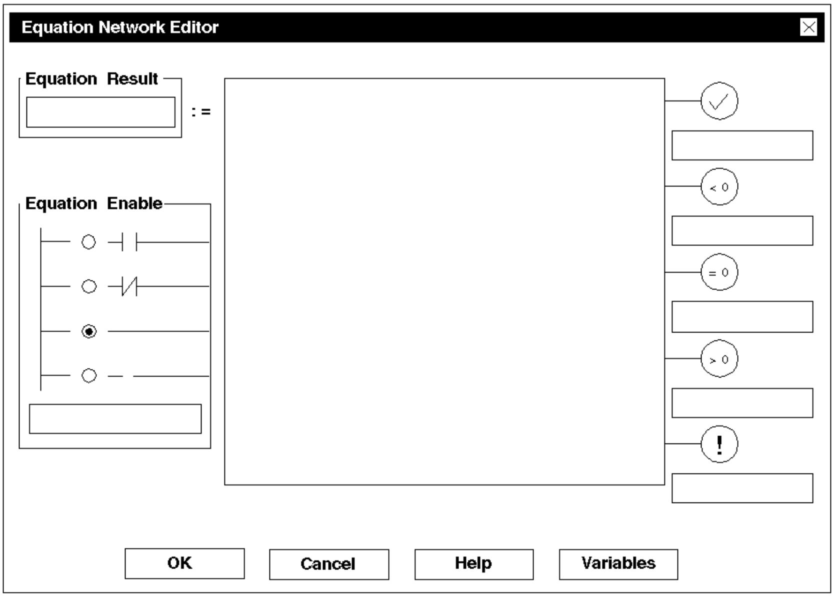
Available Menu Items

The **Networks** main menu includes two submenu entries to support equation networks: **Insert Equation** and **Append Equation**. If you page through the networks and reach the start/end of the section, you have the opportunity to insert/append a new equation network, in addition to the other choices available (insert/append ladder network, cancel, etc.).

Representation

The Ladder Logic network display changes to accommodate an initialized equation network. The row and column numbers are removed and also the grid lines are removed if they are currently being displayed.

The initial display is replaced by the figure below when you double click on the default equation body.



Equation Editing

Equation Entries

In the first column of the network, row 1 column 1, the legal equation enable entries are:

- Normally open contact (-| |-)
  When a normally open contact is entered as the first node of the network the equation is solved when the contact's referenced coil or input is ON.
- Normally closed contact (-|/|-)
  When a normally closed contact is entered as the first node of the network the equation is solved when the contact's referenced coil is OFF.
- Horizontal short (-----)
  When a horizontal short is entered as the first node of the network the equation to be solved on every scan.
  The horizontal short is used for display purposes only and is not sent to the PLC as part of the network; the absence of an enabling contact node in the network sent to the PLC indicates that the network should always be solved.
- Horizontal open (- ---)
  When a horizontal open is entered as the first node of the network the execution of the equation network is prevented.

Equation Results

Equation network can produce five possible outputs from the top five rows of the network to describe the result of the equation. You choose the outputs you want to use by assigning 0x reference numbers to them.

The outputs are displayed as coils in the last column of the equation network.

The row in which the output coils are placed determines their meanings:

- Done without error (-(√))
  When the equation passes power to the output from the top row, the equation has completed successfully without an error.
- Result < 0 (-(< 0))
  When the equation passes power to the output from the second row, the equation has completed successfully and the result is less than zero.
- Result = 0 (-(= 0))
  When the equation passes power to the output from the third row, the equation has completed successfully and the result is equal to zero.
- Result > 0 (-(> 0))
  When the equation passes power to the output from the fourth row, the equation has completed successfully and the result is greater than zero.
- Done with error (-(!))
  When the equation passes power to the output from the fifth row, the data in the equation has caused a calculation error.

Cut, Copy and Paste

Text may be pasted into the edit box of an **Equation Network Editor** dialog. These are standard Windows text operations, and are the only cut/copy/paste operations allowed within equation networks. No validation is performed at the time of a cut or paste; the equation is validated when the user decides to terminate the dialog with the **OK** button.

You can cut/copy/paste equation networks using **Network →Cut/Copy...** in which a netwotk is manipulated in its entirety.

When a network is cut or copied it may be pasted as a new equation network. In this case, "paste" means "insert new network". This is the same operation as is used with ladder networks.

Validity Check

When **OK** is selected in the **Equation Network Editor** dialog, the equation is checked for validity. If an error is detected the cursor is placed as near to the error as possible and an error message is displayed.

Syntax and Semantics

Operators

The operators are listed below in order of precedence highest to lowest. If required competing operators are evaluated left to right.

Operator Group	Operators	Description
Unary	-	Negation
	~	Ones complement
Exponentiation	**	Exponentiation
Multiply/divide	*	Multiply
	/	Divide
Add/subtract	+	Addition
	-	Subtraction
Bitwise	&	And
	-	Or
	< <	Left shift
	> >	Right shift
	^	Xor
Relations	<	Less than
	< =	Less than or equal
	=	Equal
	< >	Not equal
	= >	Greater than or equal
	>	Greater than
Conditional	?:	Test

Functions

Additionally the following functions are recognized (and predefined) in an equation:

Function	Description
ABS	Absolute value
ARCCOS	Arc Cosine
ARCSIN	Arc Sine
ARCTAN	Arc Tangent
COS	Cosine of Radians
COSD	Cosine of Degrees
EXPE	Exponential function, e** argument
FIX	Convert float to integer, presumes floating point argument
FLOAT	Convert Integer to Floating point
LN	Natural Logarithm (base e)
LOG	Common Loagarithm (base 10)
SIN	Sine of Radians
SIND	Sine of Degrees
SQRT	Square Root
TAN	Tangent of Radians
TAND	Rangent of Degrees

Equation Syntax

Equation syntax conventions:

Command	Description
[abc]	Any one of a b c
[a-z]	Any characters in the range a trough z
expr*	Zero or more expr
expr+	One or more expr

Lexical Classes

Table of lexical classes

letter	a-z A-Z
bit	0-1
octal_digit	0-7
digit	0-9
hex_digit	0-9 a-f A-F
letter_or_digit	letter | digit
identifier	letter letter_or_digit*
assignment_op	:=
relational_op	> < >= <= = <>
bitwise_op	& | ^ >> <<
add_sub_op	+ -
Mul_div_op	* /
exp_op	**
unary_op	- ~
optional_sign	+ - /*nothing*/

Constants

Constants consist of:

- binary_const 2# bit binary_const_body
- decimal_const digit decimal_const_body
- octal_const 8# octal_digit octal_const_body
- hex_const 16# hex_digit hex_const_body
- float_const mantissa exponent

Register References

reg_rvalue consists of:

discrete_rvalue	0 digit+	1 digit+	
int_reg_rvalue	3 digit+	4 digit+	6 digit+
uint_reg_rvalue	U3 digit+	U4 digit+	U6 digit+
long_reg_rvalue	L3 digit+	L4 digit+	L6 digit+
ulong_reg_rvalue	UL3 digit+	UL4 digit+	UL6 digit+
float_reg_rvalue	F3 digit+	F4 digit+	F6 dgit+

reg_lvalue consists of:

int_reg_lvalue	4 digit+	6 digit+
uint_reg_lvalue	U4 digit+	U6 digit+
long_reg_lvalue	L4 digit+	L6 digit+
ulong_reg_lvalue	UL4 digit+	UL6 digit+
float_reg_lvalue	F4 digit+	F6 dgit+

Note

Because of Concept IEC standards, placement of lexical identifiers differ between Modsoft and Concept. However, an existing Modsoft Equation is properly transformed using the Modsoft program converter.

For example a Modsoft equation

```
400100F := 400001UL + 400002U + 400003L + #23
```

becomes a Concept equation

```
%F400100 := %UL400001 + %U400002 + %L400003 +23
```

12.5 LL984 Programming Modes

LL984 Programming Modes

Direct Programming

There are two situations that determine how direct mode ladder editing is applied:

- The first is where there is no open project and you are connected to a PLC that has a valid program in it. When you select the command **Direct-mode 984LL Editor** the first program in the first segment is displayed. You can see the direct mode status at the right side of the status bar and the network window is labled **984LL Direct**.
- The second case occurs when you have a project open and you are connected to the PLC (but not **EQUAL**). When you select **Direct-mode 984LL Editor** in this case a dialog is displayed listing segments and the number of networks contained in each. Click on the segment you want click on **OK** and the **Network edit** window is displayed with a window labeled **984LL Direct**. If you have an orignal edit window it will remain on the display.

Combination Mode

Combination programming occurs when the programming panel is online. Valid program changes are immediately written to both the controller and the program database simultaneously.

DFBs (Derived Function Blocks)

13

Overview

This Chapter describes the procedure for creating DFBs (Derived Function Blocks) with help from Concept DFB.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
13.1	DFBs (Derived Function Blocks)	482
13.2	Programming and calling up a DFB	501

13.1 DFBs (Derived Function Blocks)

Overview

This section provides an overview on creating and applying DFBs (Derived Function Blocks).

What's in this Section?

This section contains the following topics:

Topic	Page
General information about DFBs (Derived Function Blocks)	483
Global / Local DFBs	485
Use of variables in DFBs	487
Combined Input/Output Variables (VARINOUT Variables)	489
Global Variables	496
Creating Context Sensitive Help (Online Help) for DFBs	499

General information about DFBs (Derived Function Blocks)

At a Glance

DFBs are created with the help of the Concept DFB software.

DFBs (Derived Function Blocks) can be used for setting both the structure and the hierarchy of a program.

In programming terms, a DFB represents a subroutine.

Meaning:
- Delivery/transfer of defined values to/from the subroutine
- Any complex program
- Nesting of one or more DFBs in a DFB
- Multiple DFB call up in the whole program, where the program code is bound only once during the whole program
- DFB specific local variables
- Initial value for variables
- freely definable Interface

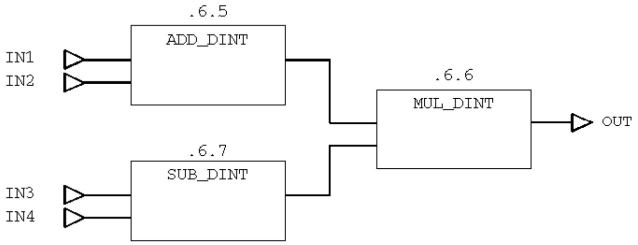
Programming languages

DFBs can be created in the Function Block language (FBD), ladder diagram (LD), instruction list (IL) and structured text (ST) programming languages.

Structure of a DFB

A DFB firstly provides an empty space, which contains a manually defined input/output and any manually programmed logic. The hierarchic structure of this logic corresponds to a project in Concept which consists of one or more sections. These sections contain the actual logic.

Internal structure of the DFB in the FBD editor:



Processing sequence

The processing sequence of the logic, the programming rules and the usable FFBs and DFBs correspond overall to those of the FBD, LD, IL and/or ST programming.

Nesting

It is possible to call up one or more already existing DFBs in a DFB, where the called up DFBs themselves can call up one or more DFBs. A DFB cannot however contain itself. A nesting depth of 5 should not be exceeded. The exact border depends, among other factors, upon parameterization (e.g. the number of DFB input/output variables) of the CPU in use and its configuration.

NOTE: If nested DFBs are used, the whole nested DFB hierarchy is not checked consistently in the DFB editor, but only the DFB on the next level. This means that, for example, with a DFB with 3 or 4 levels, the deep nested DFBs can be altered (i.e. Pin assignment), without this being apparent. In Concept, an error is not reported until project analysis.

NOTE: NEVER use diagnostic EFBs (diagnostic library) in DFBs.

Context help

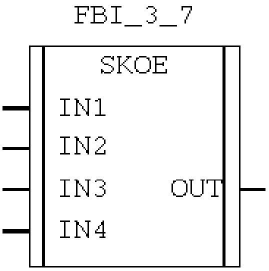
Personalized context-sensitive help (online help) can be created for DFBs (see *Creating Context Sensitive Help (Online Help) for DFBs, page 499*).

Calling up a DFB

DFBs are visually denoted in the FBD and LD editor window by double vertical lines on the DFB border. Using the command button **Despeckle...** in the properties dialog box of the DFB a document window can be opened, in which the programmed logic of the DFB can be viewed (even when it was created with IL or ST). This document window has a gray background, which denotes that the DFB in this document window cannot be edited.

DFBs are treated as Function Blocks after they are called into Concept.

Call up of the DFB in the FBD editor:



Archiving and Documentation

The archiving and documentation of a DFB is the same as with projects (see *Documentation and Archiving, page 729*).

Global / Local DFBs

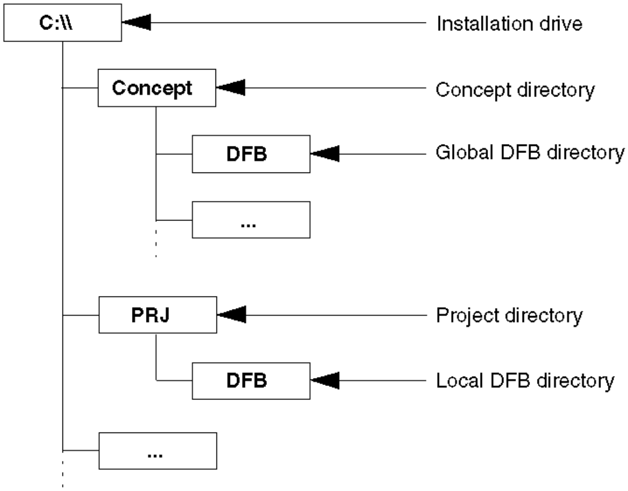
Description

Global and local DFBs differ in the locality of their directory hierarchy.

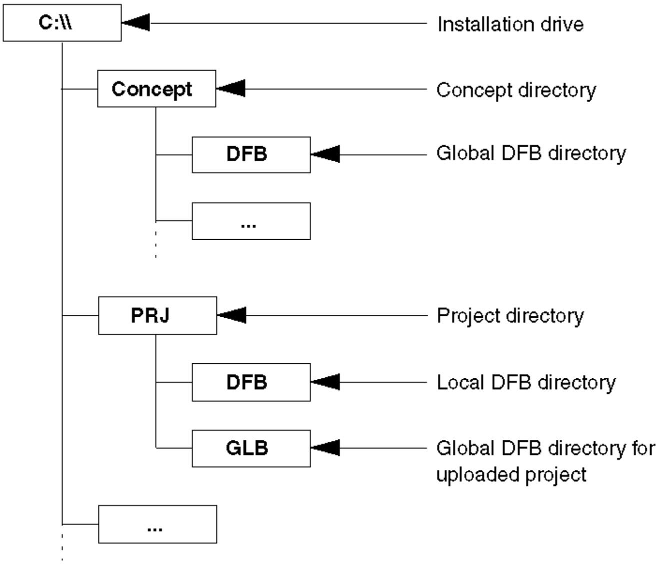
Depending on the directory or subdirectory in which the DFB is stored, it can be called up globally, i.e. within all the projects created under Concept, or locally, in a specific project.

In the *Defining the Storage of Global DFBs during Upload, page 1110* you can ensure that during the IEC upload process a GLB directory containing the global DFBs is created in the project directory. By doing this, the existing global DFBs in the **Concept** →**DFB** will not be overwritten and therefore it will not have any effect on other projects.

Directory structure without uploaded project:



Directory structure according to INI settings (**[Upload]: PreserveGlobalDFBs=1**) for uploaded projects:



If a local and a global DFB have the same name, the local DFB is given priority.

NOTE: The length of the DOS path name in which the DFBs are stored is limited to 29 characters. Care should be taken that the DFB directory does not exceed this limit.

Use of variables in DFBs

Introduction

When programming DFBs, two forms of variables are distinguished:
- Internal variables
- Formal parameters (Input/Output variables)

Internal variables

Internal variables are variables that are only used within the logic of DFBs. These variables can only be altered in Concept DFB itself. This alteration is therefore valid for all instances of this DFB.

The following are permitted as types of variables:
- Unlocated variables,
- Unlocated Multi-element variables,
- Constant variables
- Literals and
- Located variables.

NOTE: Located variables can be used if in the **IEC Extentions** dialog box you activate the **Allow located variables in DFBs** option (see also section *Global Variables, page 496*).

These variables are declared in the Variable editor *(see page 485)*.

Formal parameters

Input and output variables are required to transfer values to or from a DFB. These types of variables are called formal parameters. These variables are taken from the DFB and displayed as input/output when calling up the DFB.

In the Variable editor *(see page 485)* define the formal parameter names (the names of the inputs/outputs), the type of data and the position of the inputs/outputs (for the FBD /LD editor) on the DFB.

A maximum of 32 input and 32 output variables are possible. The width of the DFB symbol is automatically matched to the length of the name of the inputs/outputs. Input and output variables are always Unlocated variables.

An initial value can also be defined for input variables. Input variables, i.e. inputs, are always shown to the left of the DFB in the FBD/LD editor. Output variables, i.e. outputs, are always shown to the right of the DFB.

A special form of input/output variables are the VARINOUT variables *(see page 489)*.

Transfer of values during the program runtime

During program runtime, the value of the current parameters in the DFB program are passed and redistributed via the formal parameters. The value of these formal parameters are determined by the value of the current parameters, which have been linked with the corresponding DFB input/output. The current parameters can be direct addresses, located variables, unlocated variables, located multi-element variables, unlocated multi-element variables, elements from multi-element variables, constants or literals.

Through this, the same DFB type can be called up several times and each copy of the DFB assigned with individual parameters.

Exchanging positions

If all 32 possible input or output variables are occupied when creating the DFB and the exchange of the positions of 2 variables is required, a variable can be placed in position 33 in the meantime. This enables the alteration of the variable positions. However, saving a DFB with 33 input or output variables is not possible. Position 33 only serves as an auxiliary position when editing.

Combined Input/Output Variables (VARINOUT Variables)

Introduction

Combined input/output variables are a special form of input/output variables. These are also called VARINOUT variables.

Application Purpose

DFBs are often used to read a variable on input (input variables), to process it and to restate the altered values of the same variable (output variables). If the variables are structured variables and elements unaffected by the processing are also to be output again at the output, it is necessary to copy the complete variable within the DFB from the input to the output. This is also necessary when only a single element of a structured variable is processed in the DFB. To save memory and shorten the execution time, it is sensible to use VARINOUT variables in this case. This variable type can (must) be used simultaneously at DFB inputs and the associated DFB outputs.

Creating a VARINOUT variable in DFB

The following conditions must be noted when creating a VARINOUT variable:

- Like all input/output variables, VARINOUT variables are created in the Variable Editor.
- VARINOUT variables are declared twice. Once as input variables and once as output variables.
- The same formal parameter names must be used in both declarations.
- The same data types must be used in both declarations.
- The same pin positions must be used in both declarations.
- The input variable is declared first, and then the output variable.
- After confirming the declaration with **OK**, it is no longer possible to modify the input variable.

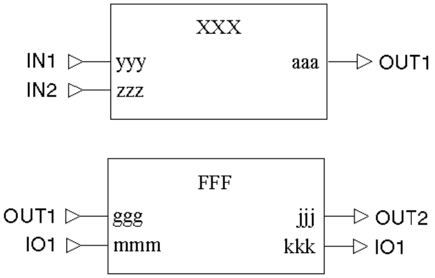
Specific Features during Creation

The following special features are to be noted when creating DFBs with VARINOUT inputs/outputs.

- If the DFB VARINOUT input has been assigned an initial value, this is not used, as it is imperative that the input is switched on.

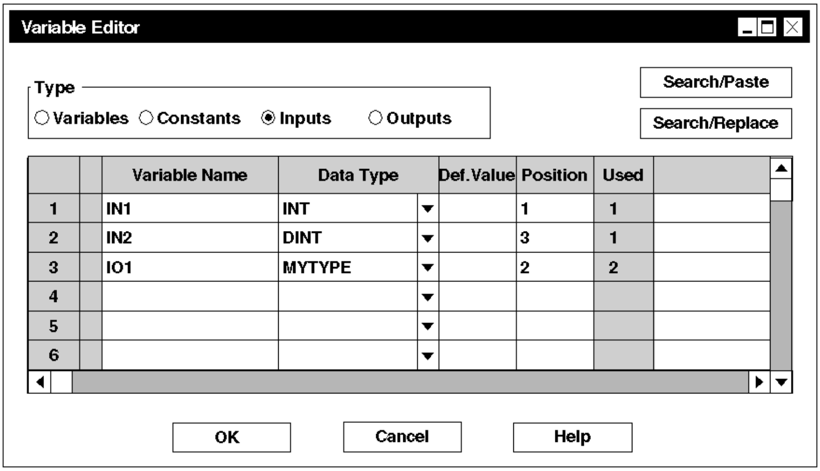
Example

DFB logic:



(* IO1 = VARINOUT variable *)

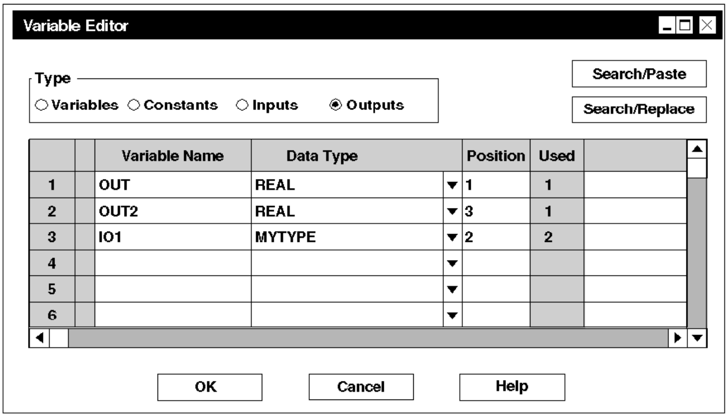
Declaration of inputs:



Type
○ Variables ○ Constants ● Inputs ○ Outputs

		Variable Name	Data Type	Def.Value	Position	Used	
1		IN1	INT		1	1	
2		IN2	DINT		3	1	
3		IO1	MYTYPE		2	2	
4							
5							
6							

Declaration of outputs:



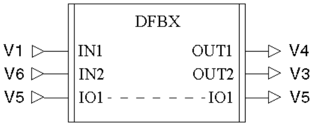
Type
○ Variables ○ Constants ○ Inputs ● Outputs

		Variable Name	Data Type	Position	Used	
1		OUT	REAL ▼	1	1	
2		OUT2	REAL ▼	3	1	
3		IO1	MYTYPE ▼	2	2	
4			▼			
5			▼			
6			▼			

Use of the DFB in FBD/LD

The DFB is invoked and used in FBD/LD editor (see also *Calling up a DFB in the FBD Function Block dialog, page 515* and *Calling up a DFB in Ladder Diagram LD, page 517*) just like any other DFB. The inputs/outputs of type VARINOUT are characterized by a dotted line.

Use of the DFB in the FBD editor:



Specific features in usage

The following special features are to be noted when using DFBs with VARINOUT inputs/outputs.

- It is essential that VARINOUT inputs/outputs are linked. Otherwise an error message appears during the section analysis.
- The same variables/variable components must be attached at the VARINOUT input and the VARINOUT output.
- No graphical links can be attached to VARINOUT inputs/outputs.
- No literals or constants can be attached to VARINOUT inputs/outputs.
- No Boolean variables can be attached to VARINOUT inputs/outputs, because this leads to problems in the code generation.
- No negations can be used at VARINOUT inputs/outputs.
- If a DFB with VARINOUT inputs/outputs is used within another DFB (nested DFBs), the VARINOUT inputs/outputs of the inner DFB can be linked to those of the outer DFB.

Use of the DFB in ST

The DFB is invoked and used in ST Editor (see also *Function Block/DFB Invocation, page 442*) like any other DFB.

Use of the DFB in the ST Editor:

```
(* Function Block declaration *)
   VAR
      Instance_Name : DFBX;
   END_VAR

(* Block invocation *)
   Instance_Name (IN1 := V1,
                  IO1 := V5,
                  IN2 := V2);

(* Assignments *)
   V4 := Instance_Name.OUT1;
   V3 := Instance_Name.OUT3;
```

The following special features are to be noted when using DFBs with VARINOUT inputs/outputs.

- It is essential that VARINOUT inputs be assigned a value on DFB invocation. Otherwise an error message will appear during the section analysis i.e. the following block invocation is not allowed, because the assignment of a value at the VARINOUT input "V5" is missing:

  ```
  Instance_Name (IN1 := V1,
                 IN2 := V2);
  ```
- VARINOUT outputs are not to be assigned a value. Otherwise an error message will appear during the section analysis i.e. the following output assignment is not allowed, because a value has been assigned at the VARINOUT output:

  ```
  V5 := Instance_Name.IO1;
  ```
- No literals or constants are to be assigned to VARINOUT inputs.
- No Boolean variables can be attached to VARINOUT inputs/outputs, because this leads to problems in the code generation.
- If a DFB with VARINOUT inputs/outputs is used within another DFB (nested DFBs), the VARINOUT inputs/outputs of the inner DFB can be linked to those of the outer DFB.

Use of the DFB in IL

The DFB is invoked and used in IL editor (see also *Use of Function Blocks and DFBs, page 372*) like any other DFB.

Use of the DFB in the IL editor:

```
(* Function Block declaration *)
   VAR
      Instance_Name : DFBX;
   END_VAR

(* Block invocation *)
   CAL Instance_Name (IN1 := V1, IO1 := V5, IN2 := V2)

(* Assignments *)
   LD Instance_Name.OUT1
   ST V4
   LD Instance_Name.OUT3
   ST V3
```

The following special features are to be noted when using DFBs with VARINOUT inputs/outputs.

- It is essential that VARINOUT inputs be assigned a value on DFB invocation. Otherwise an error message will appear during the section analysis i.e. the following block invocation is not allowed, because the assignment of a value at the VARINOUT input "V5" is missing:
  ```
  CAL Instance_Name (IN1 := V1, IN2 := V2)
  ```
- VARINOUT outputs are not to be assigned a value. Otherwise an error message will appear during the section analysis i.e. the following output assignments are not allowed, because a value has been assigned at the VARINOUT output:
  ```
  LD Instance_Name.IO1
  ST V5
  ```
- No literals or constants are to be assigned to VARINOUT inputs.
- No Boolean variables can be attached to VARINOUT inputs/outputs, because this leads to problems in the code generation.
- If a DFB with VARINOUT inputs/outputs is used within another DFB (nested DFBs), the VARINOUT inputs/outputs of the inner DFB can be linked to those of the outer DFB.

Special features when modifying

There are 3 general possibilities for modifying VARINOUT variables:

- Modify existing VARINOUT variables:
  - Rename the variables
  - Change the data type
  - Change the pin position
- Two existing variables can be joined in one VARINOUT variable
- Split a VARINOUT variable into two variables

Change existing VARINOUT variables

To change (rename, change data type, change pin position) existing VARINOUT variables, proceed as follows:

Step	Action
1	Open the Variable Editor (**F8**).
2	Select the **Outputs** option.
3	Implement the required changes. Response: The changes are automatically transferred to the input variable.
4	Confirm the changes with **OK**.

Join variables to VARINOUT variable

To join two variables to a VARINOUT variable, perform the following steps:

Step	Action
1	Open the Variable Editor (**F8**).
2	Select the **Inputs** option.
3	Create a new input variable (e.g. INOUT1).
4	Select the **Outputs** option.
5	Create a new output variable with the same name (e.g. INOUT1), data type and pin position as the input variable.
6	Confirm the changes with **OK**.
7	Replace all uses of the input and output variable with the VARINOUT variable in your program.
8	Open the Variable Editor (**F8**) and delete the now redundant input and output variable.

Splitting VARINOUT variable

To split a VARINOUT variable into two variables, proceed as follows:

Step	Action
1	Open the Variable Editor (**F8**).
2	Select the **Inputs** option.
3	Create a new input variable (e.g. IN1).
4	Select the **Outputs** option.
5	Create a new output variable (e.g. OUT1).
6	Confirm the changes with **OK**.
7	Replace all usages of the VARINOUT variable with the input and output variables in your program.
8	Open the Variable Editor (**F8**) and delete the now redundant VARINOUT variable.

Global Variables

Introduction

Global variables are located variables which are declared in Concept-DFB and Concept.

Global variables in DFBs can only be declared if the **Allow Located Variables in DFBs** check box is activated in the **IEC Extensions** dialog box. Then the the **Address** column is available in the DFB Variable Editor, i.e. located variables can now be declared.

Global validity of the variables is defined as soon as the DFB is used in the project and the respective located variables are declared in the Concept Variable Editor. When declaring the variables, make sure that the same name, address and data type is used as in the DFB Variable Editor. All reference ranges can be used (0x, 1x, 3x and 4x).

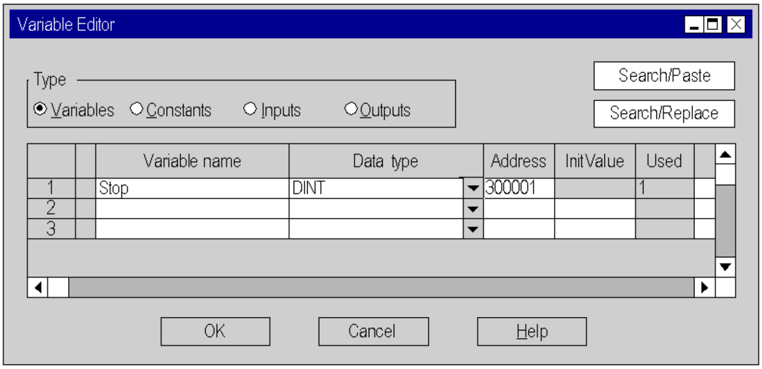
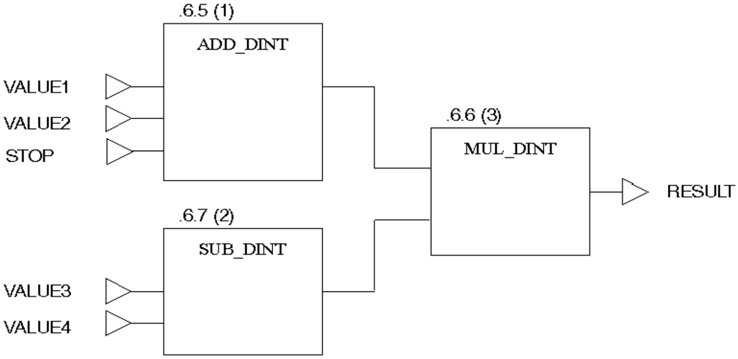
Declaration errors are found and error messages are given when the program is analyzed (**Project →Analyze program**). If global validity is recognized, the global variables are shown with a gray background in the Concept Variable Editor and are write protected in Concept. That means global variables canonly be changed in the DFB Variable Editor. Then the declaration for the changed variables must be updated in the Concept Variable Editor to restore global validity.

NOTE: If inconsistencies are found between the declaration of global variables in the DFB and the program when analyzing the program (e.g. the address is declared differently), the program cannot be downloaded to the PLC.

Execution in Concept-DFB

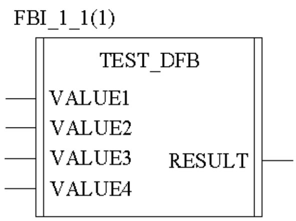
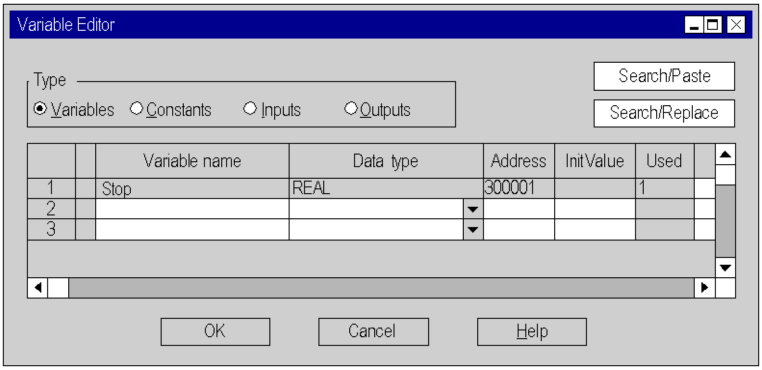
To create global variables in DFB, carry out the following steps in Concept-DFB:

Step	Action
1	Close Concept and Start Concept-DFB.
2	Select **Options →Preferences →IEC Extensions...**, and activate the check box **Allow Located Variables in DFBs**.
3	Create a DFB (see section *Creating the DFB, page 503*).
4	Create the logic (example: see section *Creating the Logic in FBD Function Block Language, page 504*).

Step	Action
5	Select **Project** →**Variable declarations**. To declare the located variables, activate the **Variables** option button. **Note:** All reference ranges can be used (0x, 1x, 3x and 4x) for addressing.  Variable Editor Type: ● Variables ○ Constants ○ Inputs ○ Outputs Variable name / Data type / Address / InitValue / Used 1 Stop DINT 300001 1 2 3
6	Now re-activate the selection mode with **Objects** →**Select mode** and double-click on one of the unconnected inputs. **Result:** The **Connect FFB** dialog box is opened, where you can assign a current parameter to the input.
7	In **Connect with**, activate the **Variable** option button.
8	Open the variable editor using he **Variable declaration...** command button. Then select the unlocated variable (STOP) and click **OK**. **Result:** The selected variable is transferred to the text box in the **Connect FFB** dialog box.
9	With **OK**, the variable (STOP) is assigned to the selected input on the module.  .6.5 (1) ADD_DINT VALUE1 VALUE2 STOP .6.6 (3) MUL_DINT RESULT .6.7 (2) SUB_DINT VALUE3 VALUE4
10	Save the DFB using the menu command **File** →**Save**.

Execution in Concept

To create global variables in DFB, carry out the following steps in Concept:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Close the Concept DFB and Start Concept.</td></tr>
<tr><td>2</td><td>Call the DFB (example: see section <i>Calling up a DFB in the FBD Function Block dialog, page 515</i>).

</td></tr>
<tr><td>3</td><td>Select <b>Project</b> →<b>Variable declarations...</b>. To declare the located variables (STOP), activate the <b>Variables</b> option button.</td></tr>
<tr><td>4</td><td>Transfer the variable names, data type and the address of the located variables, exactly as they were declared in the Concept-DFB variable editor.</td></tr>
<tr><td>5</td><td>Analzye the program using <b>Project</b> →<b>Analyze program</b>.
<b>Result:</b> The <b>Messages</b> window is opened and shows that the global variable "STOP" was found in the DFB.
The global validity of the variable is recognized, therefore it is shown with a gray background in the Concept Variable Editor.


Type
● Variables ○ Constants ○ Inputs ○ Outputs
<table>
<tr><th></th><th></th><th>Variable name</th><th>Data type</th><th>Address</th><th>InitValue</th><th>Used</th></tr>
<tr><td>1</td><td></td><td>Stop</td><td>REAL</td><td>300001</td><td></td><td>1</td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td>6</td><td>In the DFB Editor, you can open the <b>Function Block</b> dialog box by double-clicking on the DFB. Using the <b>Refine...</b> command button, open a document window with the inner logic of the DFB (the global variable STOP is also shown here).</td></tr>
</table>

Creating Context Sensitive Help (Online Help) for DFBs

Introduction

In Concept, help is provided for each EFB, which can be invoked according to the context (the **Help on Type** command key in the EFBs properties dialog). There are of course no corresponding help texts in Concept for the DFBs created by you.

You can, however, create your own help for each DFB, which can be invoked in Concept with **Help on Type**.

File Format:

You can create your help in the following file formats:

- **.chm** (Microsoft Windows compiled HTML help file)
- **.doc** (Microsoft Word format)
- **.htm** (Hypertext Markup Language)
- **.hlp** (Microsoft Windows help file (16- or 32-Bit Format))
- **.pdf** (Adobe Portable Document Format
- **.rtf** (Microsoft Rich Text Format)
- **.txt** (Plain ASCII Text-Format)

Name

The name of the help file must be exactly the same as the name of the DFB (e.g. SKOE.ext)

The only exceptions are standardized DFB names (e.g. SKOE_BOOL, SKOE_REAL etc.) In these cases the help file name is the DFB name without the datatype extension (e.g. DFB name) SKOE_BOOL has the help file SKOE.ext).

Directory

The help file can be stored in the following directories:

- Concept directory
- Concept Help directory (if defined in the file Concept.ini, see readme)
- Global DFB directory
- Local DFB directory

Invoking the Help File

Concept carries out the following procedure to invoke the help file:

Phase	Description
1	Search for the help file **DFBName.ext** in the local DFB-directory. The help file is searched for in the following sequence: ● .hlp ● .chm ● .htm ● .rtf ● .doc ● .txt ● .pdf **Result:** If the search result is positive the help file will be displayed, otherwise it will continue with phase 2.
2	Search for the help file **DFBName.ext** in the global DFB-directory. For the order, see phase 1. **Result:** If the search result is positive the help file will be displayed, otherwise it will continue with phase 3.
3	Search for the help file **DFBName.ext** in the Concept-directory or Concept-Help directory. For the order, see phase 1. **Result:** If the search result is positive the help file will be displayed, otherwise it will continue with phase 4.
4	Display of the comment created in Concept DFB with **Project** →**Properties**.

13.2 Programming and calling up a DFB

Overview

This section describes programming and calling up a DFB.

What's in this Section?

This section contains the following topics:

Topic	Page
At a Glance	502
Creating the DFB	503
Creating the Logic in FBD Function Block Language	504
Creating the Logic in LD Ladder Diagram	507
Creating the Logic in IL Instruction List	511
Creating the Logic in ST Structured Text	513
Calling up a DFB in the FBD Function Block dialog	515
Calling up a DFB in Ladder Diagram LD	517
Calling up a DFB in the IL instruction list	519
Calling up a DFB in structured text ST	520

At a Glance

At a Glance

Programming and calling up a DFB is divided into 3 main steps:

Step	Action
1	Occupying the DFB *(see page 503)*
2	Creating the logic in: ● Function Block language (FBD) *(see page 504)* ● Ladder diagram (LD) *(see page 507)* ● Instruction list (IL) *(see page 511)* ● Structured text (ST) *(see page 513)*
3	Calling up the DFB in: ● Function Block language (FBD) *(see page 515)* ● Ladder diagram (LD) *(see page 517)* ● Instruction list (IL) *(see page 519)* ● Structured text (ST) *(see page 520)*

Creating the DFB

Description

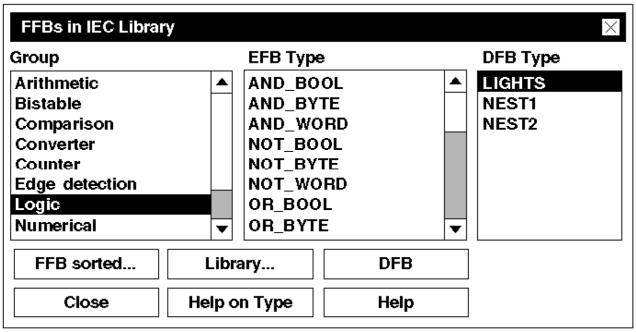
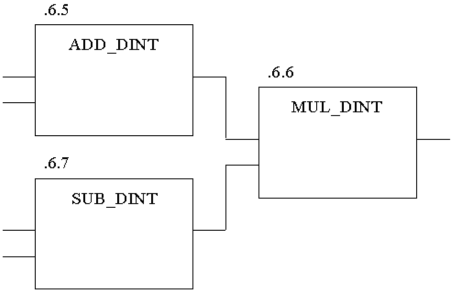
The procedure for creating the DFB is as follows:

Step	Action
1	Close Concept and start Concept DFB.
2	Create a new DFB using the menu command **Data file →New DFB**. **Reaction:** The name now appears on the title bar:**[untitled]**.
3	Using the menu command **Data file →New section...** , generate a new section and enter a section name. The section name (max. 32 characters) must be clear throughout the DFB, and is not case-sensitive. If the section name entered already exists, a warning is given and another name must be chosen. The section name must correspond to the IEC Name conventions, otherwise an error message appears. **Note:** In accordance with IEC1131-3, only letters are permitted as the first character of names. If, however numbers are required as the first character, this can be enabled using the menu command **Options →Pre-settings →IEC Expansions... →IEC Expansions →Enable leading figures in identifiers** .
4	Select a programming language for the section: ● Function Block language (FBD) *(see page 504)* ● Ladder diagram (LD) *(see page 507)* ● Instruction list (IL) *(see page 511)* ● Structured text (ST) *(see page 513)*
5	The menu command **Project →Properties** can be used to generate a comment about the DFB. **Reaction:** This comment can be shown in Concept in the DFB properties box with the command button **Help for type**.
6	Save the DFB with the menu command **Data file →Save DFB**. **Reaction:** The first time the Save is used, the **Save as** dialog box opens – specify the DFB name and directory where it is to be saved here.
7	Select the directory to be occupied by the DFB. Attention should be paid to the difference between global and local DFBs (see also *Global / Local DFBs, page 485*).
8	Enter the DFB name (max. 8 characters, always with the .DFB extension). The name must be clear throughout the directory, and is not case-sensitive. If the section name entered already exists, a warning is given and another name must be chosen.

Creating the Logic in FBD Function Block Language

Description

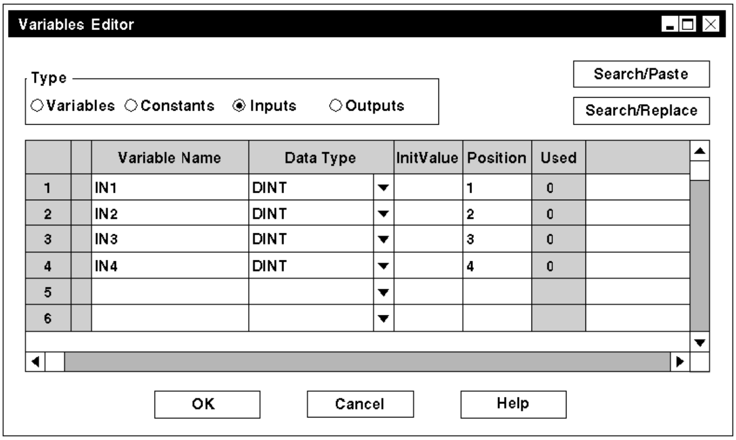
The procedure for creating the logic in FBD function block language is as follows:

Step	Action
1	To insert an FFB into the section, select the **Objects** →**Select FFB...** menu command. **Result:** The FFB dialog box from the library is opened.  FFBs in IEC Library Group Arithmetic Bistable Comparison Converter Counter Edge detection Logic Numerical EFB Type AND_BOOL AND_BYTE AND_WORD NOT_BOOL NOT_BYTE NOT_WORD OR_BOOL OR_BYTE DFB Type LIGHTS NEST1 NEST2 FFB sorted... Library... DFB Close Help on Type Help
2	In this dialog box you can select a library and an FFB from it by using the **Library...** command button. You can, however, also display the DFBs that you created and select one of them using the **DFB** command button.
3	Place the selected FFB in the section.
4	When all FFBs have been positioned, close the dialog box with **OK**
5	Activate select mode with **Objects** →**Select Mode**, click on the FFB and move the FFBs to the desired position.
6	Activate the link mode with **Objects** →**Link** and connect the FFBs. **For example:** 

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>7</td><td>

Activate the Variables Editor with**Project** →**Variable Editor** to declare the DFB variables and inputs/outputs (formal parameters).

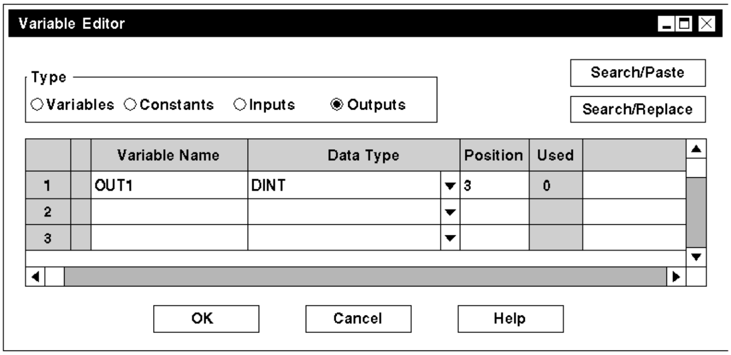
Example (inputs):



Type ○ Variables ○ Constants ● Inputs ○ Outputs

<table>
<tr><th></th><th></th><th>Variable Name</th><th>Data Type</th><th>InitValue</th><th>Position</th><th>Used</th><th></th></tr>
<tr><td>1</td><td></td><td>IN1</td><td>DINT</td><td></td><td>1</td><td>0</td><td></td></tr>
<tr><td>2</td><td></td><td>IN2</td><td>DINT</td><td></td><td>2</td><td>0</td><td></td></tr>
<tr><td>3</td><td></td><td>IN3</td><td>DINT</td><td></td><td>3</td><td>0</td><td></td></tr>
<tr><td>4</td><td></td><td>IN4</td><td>DINT</td><td></td><td>4</td><td>0</td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

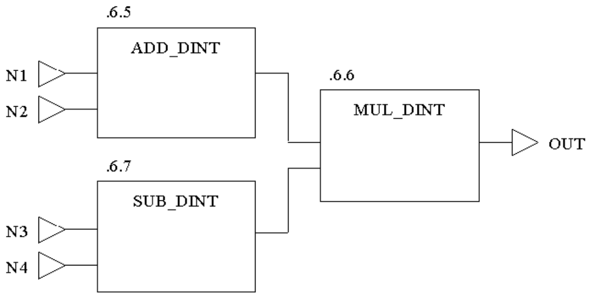
Example (outputs):



Type ○ Variables ○ Constants ○ Inputs ● Outputs

<table>
<tr><th></th><th></th><th>Variable Name</th><th>Data Type</th><th>Position</th><th>Used</th><th></th></tr>
<tr><td>1</td><td></td><td>OUT1</td><td>DINT</td><td>3</td><td>0</td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

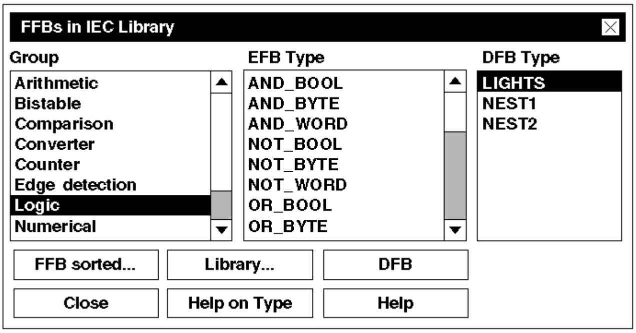
</td></tr>
</table>

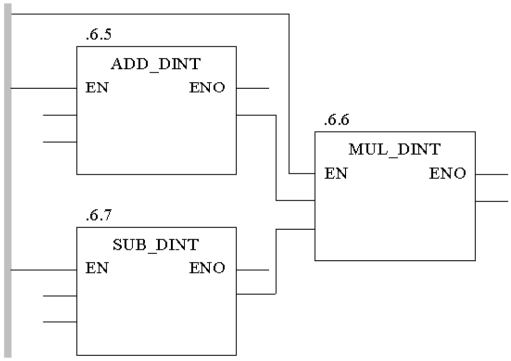
Step	Action
8	Then re-activate the select mode with **Objects** →**Select Mode** and double-click on one of the unconnected inputs/outputs. **Result:** The **Connect FFB** dialog box opens, in which you can allocate a current parameter to the input/output.
9	Back up the DFB with the **File** →**Save** menu command. **For example:**  .6.5 ADD_DINT N1 N2 .6.6 MUL_DINT OUT .6.7 SUB_DINT N3 N4

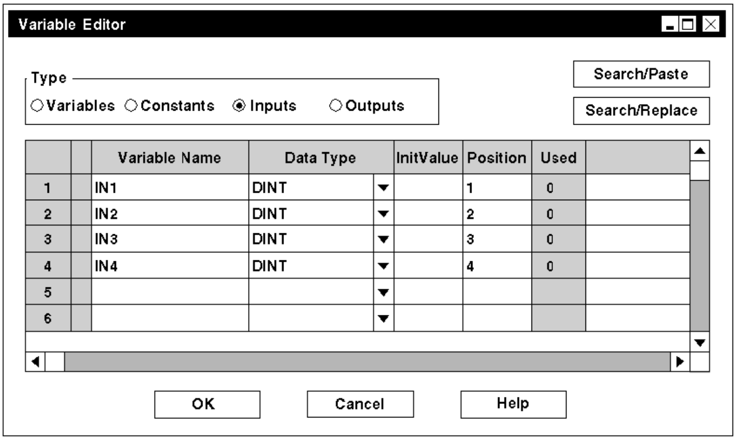
Creating the Logic in LD Ladder Diagram

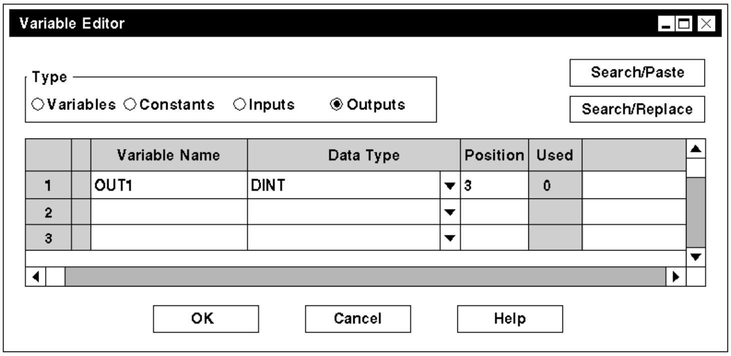
Description

The procedure for creating the logic in LD ladder diagram is as follows:

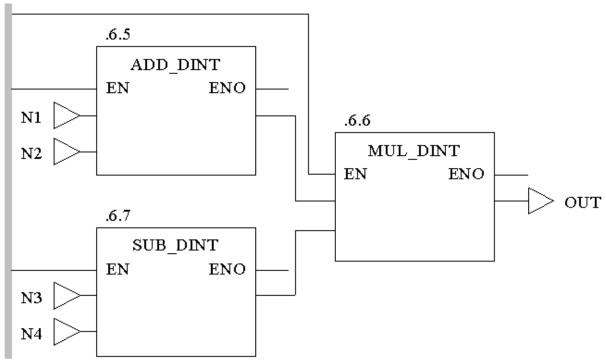
Step	Action
1	To insert a contact or coil in the section, open the **Objects** main menu and select the desired contact or coil. Contacts and coils can also be selected using the tool bar. Place the contact or coil in the section.
2	To insert an FFB into the section, select the **Objects** →**Select FFB...** menu command. **Result:** The **FFBs from Library** dialog box is opened.  FFBs in IEC Library Group Arithmetic Bistable Comparison Converter Counter Edge detection Logic Numerical EFB Type AND_BOOL AND_BYTE AND_WORD NOT_BOOL NOT_BYTE NOT_WORD OR_BOOL OR_BYTE DFB Type LIGHTS NEST1 NEST2 FFB sorted... Library... DFB Close Help on Type Help
3	In this dialog box you can select a library and an FFB from it by using the **Library...** command button. You can, however, also display the DFBs that you created and select one of them using the **DFB** command button.
4	Place the selected FFB in the section.
5	When all FFBs have been positioned, close the dialog box with **OK**
6	Activate select mode using **Objects** →**Select Mode**, and move the contacts, coils and FFBs to the required position.

Step	Action
7	Activate link mode with **Objects** →**Link**, and connect the contacts, coils and FFBs. Connect the contacts, FFBs and the left power rail. **For example:**  .6.5 ADD_DINT EN ENO .6.6 MUL_DINT EN ENO .6.7 SUB_DINT EN ENO

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>8</td><td>Activate the Variables Editor with<b>Project</b> →<b>Variable Editor</b> to declare the DFB variables and inputs/outputs (formal parameters).
<b>Example (inputs):</b>


Type: ○ Variables ○ Constants ● Inputs ○ Outputs
<table>
<tr><th></th><th></th><th>Variable Name</th><th>Data Type</th><th>InitValue</th><th>Position</th><th>Used</th><th></th></tr>
<tr><td>1</td><td></td><td>IN1</td><td>DINT</td><td></td><td>1</td><td>0</td><td></td></tr>
<tr><td>2</td><td></td><td>IN2</td><td>DINT</td><td></td><td>2</td><td>0</td><td></td></tr>
<tr><td>3</td><td></td><td>IN3</td><td>DINT</td><td></td><td>3</td><td>0</td><td></td></tr>
<tr><td>4</td><td></td><td>IN4</td><td>DINT</td><td></td><td>4</td><td>0</td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
<b>Example (outputs):</b>


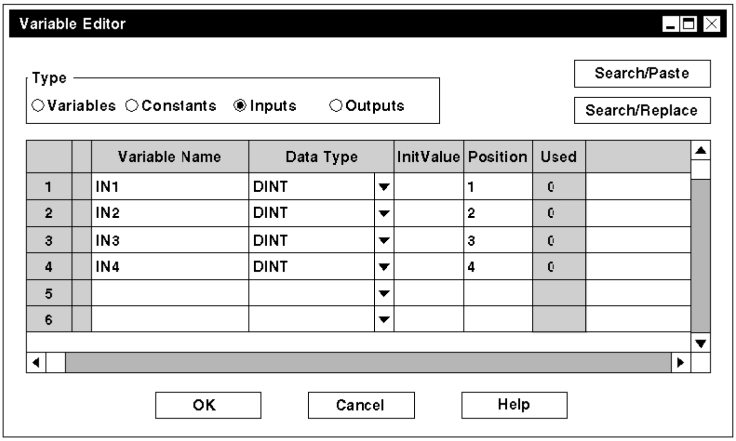
Type: ○ Variables ○ Constants ○ Inputs ● Outputs
<table>
<tr><th></th><th></th><th>Variable Name</th><th>Data Type</th><th>Position</th><th>Used</th><th></th></tr>
<tr><td>1</td><td></td><td>OUT1</td><td>DINT</td><td>3</td><td>0</td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>9</td><td>Then re-activate select mode with <b>Objects</b> →<b>Select mode</b>, and double-click on a contact or coil.
<b>Result:</b> The <b>Properties: LD Objects</b> dialog box is opened, in which you can allocate an actual parameter to the contact/coil.</td></tr>
</table>

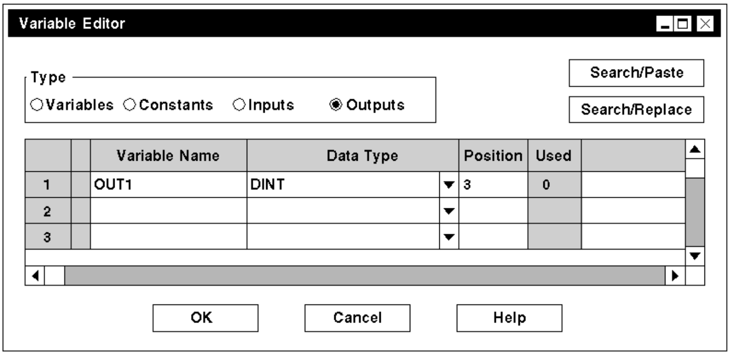
Step	Action
10	To connect the FFB input/outputs to the current parameters, double-click on one of the unconnected input/outputs. **Result:** The Connect FFB dialog box is opened, in which you can allocate a current parameter to the input/output.
11	Back up the DFB with the **File** →**Save** menu command. **For example:**  .6.5 ADD_DINT EN ENO N1 N2 .6.6 MUL_DINT EN ENO OUT .6.7 SUB_DINT EN ENO N3 N4

Creating the Logic in IL Instruction List

Description

The procedure for creating the logic in Instruction List (IL) is as follows:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Declare the function block and DFBs to be used using VAREND_VAR.
<b>Note:</b> Functions do not have to be declared:
<b>Example:</b>
VAR
    CLOCK : CLOCK_DINT ;
END_VAR</td></tr>
<tr><td>2</td><td>Declare the variables and their initial value in the Variable Editor.
<b>Example (inputs):</b>


Type
○ Variables ○ Constants ● Inputs ○ Outputs
<table>
<tr><th></th><th></th><th>Variable Name</th><th>Data Type</th><th>InitValue</th><th>Position</th><th>Used</th></tr>
<tr><td>1</td><td></td><td>IN1</td><td>DINT</td><td></td><td>1</td><td>0</td></tr>
<tr><td>2</td><td></td><td>IN2</td><td>DINT</td><td></td><td>2</td><td>0</td></tr>
<tr><td>3</td><td></td><td>IN3</td><td>DINT</td><td></td><td>3</td><td>0</td></tr>
<tr><td>4</td><td></td><td>IN4</td><td>DINT</td><td></td><td>4</td><td>0</td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
<b>Example (outputs):</b>


Type
○ Variables ○ Constants ○ Inputs ● Outputs
<table>
<tr><th></th><th></th><th>Variable Name</th><th>Data Type</th><th>Position</th><th>Used</th></tr>
<tr><td>1</td><td></td><td>OUT1</td><td>DINT</td><td>3</td><td>0</td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
</table>

Step	Action

3	Create your program's logic.
	For example:
	`LD IN1`
	`ADD IN2`
	`MUL (`
	`LD IN3`
	`SUB IN4`
	`)`
	`ST OUT`

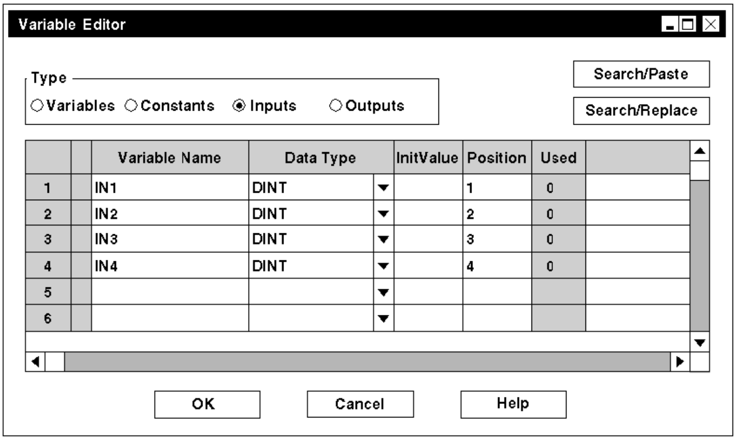
4	Back up the section with the **File →Save Project** menu command.

Creating the Logic in ST Structured Text

Description

The procedure for creating the logic in ST structured text is as follows:

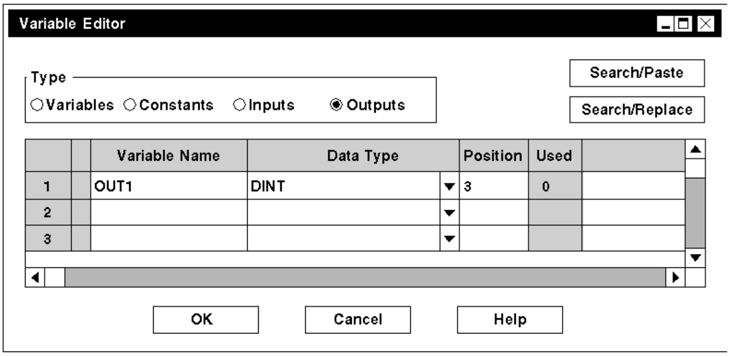
<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Declare the function block and DFBs to be used using VAREND_VAR.
<b>Note:</b> Functions do not have to be declared:
<b>Example:</b>
<pre>
VAR
   CLOCK : CLOCK_DINT ;
END_VAR
</pre></td></tr>
<tr><td>2</td><td>Declare the variables and their initial value in the Variable Editor.
<b>Example (inputs):</b>



Type
○ Variables ○ Constants ● Inputs ○ Outputs

<table>
<tr><th></th><th></th><th>Variable Name</th><th>Data Type</th><th>InitValue</th><th>Position</th><th>Used</th><th></th></tr>
<tr><td>1</td><td></td><td>IN1</td><td>DINT</td><td></td><td>1</td><td>0</td><td></td></tr>
<tr><td>2</td><td></td><td>IN2</td><td>DINT</td><td></td><td>2</td><td>0</td><td></td></tr>
<tr><td>3</td><td></td><td>IN3</td><td>DINT</td><td></td><td>3</td><td>0</td><td></td></tr>
<tr><td>4</td><td></td><td>IN4</td><td>DINT</td><td></td><td>4</td><td>0</td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

<b>Example (outputs):</b>



Type
○ Variables ○ Constants ○ Inputs ● Outputs

<table>
<tr><th></th><th></th><th>Variable Name</th><th>Data Type</th><th>Position</th><th>Used</th><th></th></tr>
<tr><td>1</td><td></td><td>OUT1</td><td>DINT</td><td>3</td><td>0</td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
</table>

Step	Action
3	Create your program's logic. **For example:** `OUT := (IN1 + IN2) * (IN3 - IN4)`
4	Back up the section with the **File** →**Save Project** menu command.

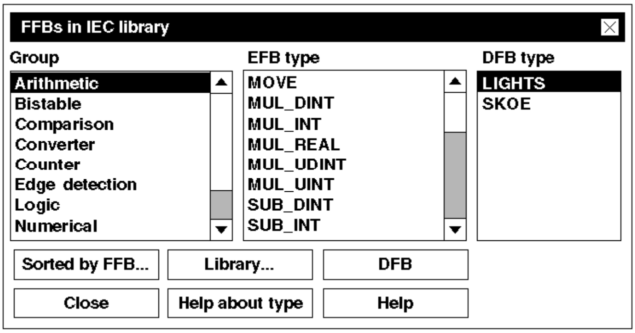
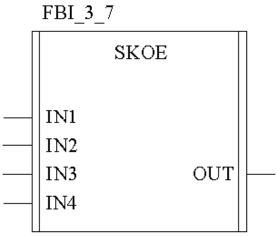
Calling up a DFB in the FBD Function Block dialog

Note

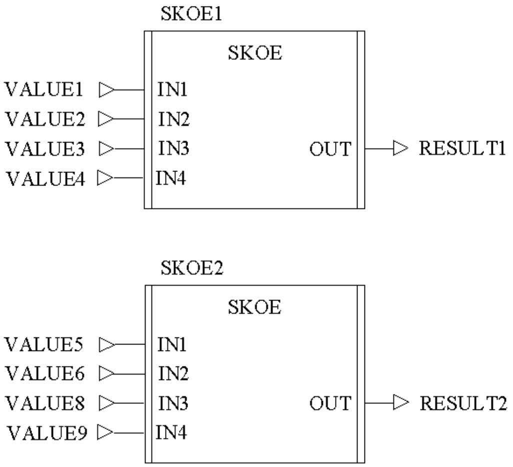
When a DFB is called up, it is not significant which program languages it was created in. The DFB can be called up in all the IEC sections.

Description

The procedure for calling up a DFB in the FBD Function Block dialog is as follows:

Step	Action
1	Close the Concept DFB and start Concept.
2	Open or create a project and open or create a section.
3	As with an EFB, the DFB is called up using the command button: **Objects** → **Select FFB...**. **Reaction:** The dialog box **FFBs from library** is opened.
4	Press the **DFB** command button to display the global and local DFBs. **For example:**  FFBs in IEC library Group Arithmetic Bistable Comparison Converter Counter Edge detection Logic Numerical EFB type MOVE MUL_DINT MUL_INT MUL_REAL MUL_UDINT MUL_UINT SUB_DINT SUB_INT DFB type LIGHTS SKOE Sorted by FFB... Library... DFB Close Help about type Help
5	Now click on the desired DFB in the list, and position it in the Editor window. **For example:** 

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>6</td><td>Double-clicking on the DFB opens the <b>Properties: Derived Function Block</b> dialog box, where the <b>Refine...</b> command button can be used to open a document window with the internal DFB logic. The gray background indicates that the DFB cannot be edited in this document window.</td></tr>
<tr><td>7</td><td>Now only the actual parameter needs to be defined. This is performed in a way corresponding to the normal EFB link using the <b>Link FFB</b> dialog box (double-click on the inputs/outputs to be parametered.
<b>For example:</b>



<b>Reaction:</b> As is clear from the example, two different sets of parameters are used in the DFB calls 1 and 2. The formal parameters are the same in both calls because the program code of the DFB is only occupied once.</td></tr>
</table>

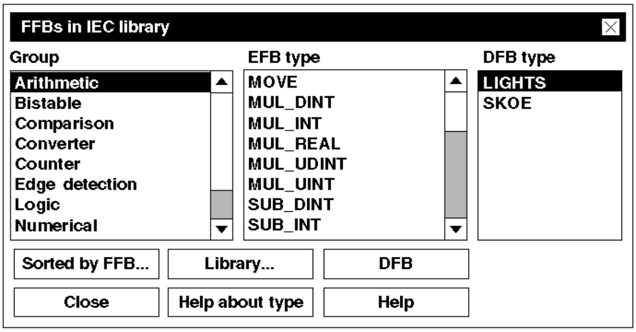
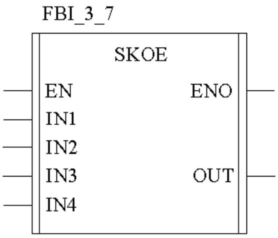
Calling up a DFB in Ladder Diagram LD

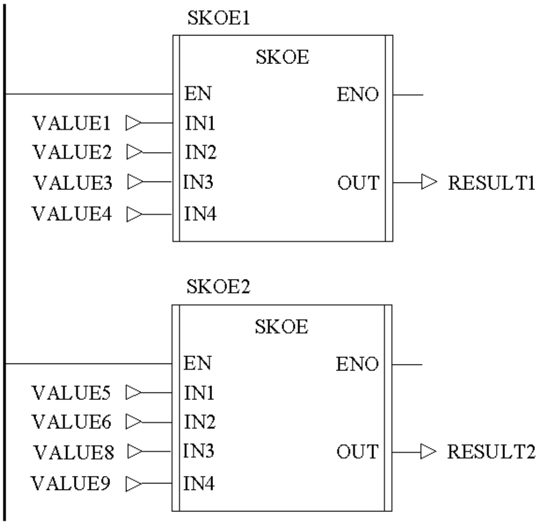
Note

When a DFB is called up, it is not significant which program languages it was created in. The DFB can be called up in all the IEC sections.

Description

To call up a DFB in Ladder Diagram LD, do the following:

Step	Action
1	Close the Concept DFB and start Concept.
2	Open or create a project and open or create a section.
3	As with an EFB, the DFB is called up using the command button: **Objects** → **Select FFB...**. **Reaction:** The dialog box **FFBs from library** is opened.
4	Press the **DFB** command button to display the global and local DFBs. **For example:**  FFBs in IEC library Group: Arithmetic (selected), Bistable, Comparison, Converter, Counter, Edge detection, Logic, Numerical EFB type: MOVE, MUL_DINT, MUL_INT, MUL_REAL, MUL_UDINT, MUL_UINT, SUB_DINT, SUB_INT DFB type: LIGHTS (selected), SKOE Sorted by FFB... Library... DFB Close Help about type Help
5	Now click on the DFB required in the list, and position it in the Editor window. **For example:**  FBI_3_7 SKOE EN ENO IN1 IN2 IN3 OUT IN4

Step	Action
6	Double-clicking on the DFB opens the **Properties: Derived Function Block** dialog box, where the **Refine...** command button can be used to open a document window with the internal DFB logic. The gray background indicates that the DFB cannot be edited in this document window.
7	Use the left power rail to link the EN input.
8	Now only the actual parameter needs to be defined. This is performed in a way corresponding to the normal EFB link using the **Link FFB** dialog box (double-click on the inputs/outputs to be parametered. **For example:**  SKOE1 SKOE EN ENO VALUE1 IN1 VALUE2 IN2 VALUE3 IN3 OUT RESULT1 VALUE4 IN4 SKOE2 SKOE EN ENO VALUE5 IN1 VALUE6 IN2 VALUE8 IN3 OUT RESULT2 VALUE9 IN4 **Reaction:** As is clear from the example, two different sets of parameters are used in the DFB call 1 and 2. The formal parameters are the same in both calls because the program code of the DFB is only occupied once.

Calling up a DFB in the IL instruction list

Note

When a DFB is called up, it is not significant which program languages it was created in. The DFB can be called up in all the IEC sections.

Description

To call up a DFB in the IL instruction list, do the following:

Step	Action
1	Close the Concept DFB and start Concept.
2	Open or create a project and open or create a section.

3	Calling up a DFB in the IL is performed like Calling up a Function Block *(see page 372)*.
	For example:
	`VAR`
	`SKOE1, SKOE2 : SKOE;      (* Instancing the DFBs *)`
	`END_VAR`
	`CAL SKOE1(IN1:=VALUE1,IN2:=VALUE2,IN3:=VALUE3,IN4:=VALUE4)`
	`LD SKOE1.out              (* DFB Call 1 *)`
	`ST RESULT1`
	`CAL SKOE2(IN1:=VALUE5,IN2:=VALUE6,IN3:=VALUE7,IN8:=VALUE4)`
	`LD SKOE2.out              (* DFB Call 2 *)`
	`ST RESULT2`
	Reaction: As is clear from the example, two different sets of parameters are used in the DFB calls 1 and 2. The formal parameters are the same in both calls because the program code of the DFB is only occupied once.

Calling up a DFB in structured text ST

Note

When a DFB is called up, it is not significant which program languages it was created in. The DFB can be called up in all the IEC sections.

Description

The procedure for calling up a DFB in structured text ST is as follows:

Step	Action
1	Close the Concept DFB and start Concept.
2	Open or create a project and open or create a section.

3	Calling up a DFB in the ST is performed like Calling up a Function Block *(see page 442)*.
	For example:
	`VAR`
	`SKOE1, SKOE2 : SKOE;      (* Instancing the DFBs *)`
	`END_VAR`
	`SKOE1(IN1:=VALUE1, IN2:=VALUE2, IN3:=VALUE3, IN4:=VALUE4);`
	`RESULT1:=SKOE1.OUT ;             (* DFB Call 1 *)`
	`SKOE2(IN1:=VALUE5, IN2:=VALUE6, IN3:=VALUE7, IN4:=VALUE8);`
	`RESULT2:=SKOE2.OUT ;             (* DFB Call 2 *)`
	Reaction: As is clear from the example, two different sets of parameters are used in the DFB calls 1 and 2. The formal parameters are the same in both calls because the program code of the DFB is only occupied once.

Macros

14

Overview

This Chapter describes the procedure for creating macros with help from Concept DFB.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
14.1	Macro	522
14.2	Programming and calling up a macro	531

14.1 Macro

Overview

This section provides an overview on creating and applying macros.

What's in this Section?

This section contains the following topics:

Topic	Page
Macros: general	523
Global / Local Macros	525
Exchange marking	527
Creating Context Sensitive Help (Online Help) for Macros	529

Macros: general

At a Glance

Macros are used to duplicate frequently used sections and networks (including their logic, variables and variable declaration).

Creating macros

Macros are created with the help of the Concept DFB software.

Programming languages

Macros can only be created in the FBD and LD programming languages.

Properties

Macros have the following properties:

- Macros only contain one section.
- Macros can contain a section of any complexity.
- From the point of view of program technology, there is no difference between an instanced macro, i.e. a macro inserted into a section and a conventionally created section.
- It is possible to call up DFBs in a macro.
- It is possible to declare macro-specific variables for the macro.
- It is possible to use data structures specific to the macro
- Automatic transfer of the variables declared in the macro.
- Initial values are possible for the macro variables.
- It is possible to instance a macro many times in the entire program with different variables.
- The name of the section, variable names and data structure names can contain up to 10 different exchange marks (@0 to @9).

Hierarchic structure

The hierarchic structure of a macro corresponds to a project in Concept which consists of only one section. This section contains the actual logic.

Context help

Personalised context-sensitive help (online help) can be generated for macros (see *Creating Context Sensitive Help (Online Help) for Macros, page 529*).

Processing sequence

The processing sequence of the logic, the programming rules and the usable FFBs and DFBs correspond overall to those of the FBD or LD programming.

Calling up a macro

A macro can be called up from SFC, FBD and LD sections.

There is a fundamental difference here:

- Call from an SFC Section
  When a macro is called up (instanced) from an SFC section (e.g. as a network for the action variable), a new FBD/LD section containing only the macro's logic is automatically created
- **Calling up an FBD/LD section**
  When a macro is called up from an FBD or LD section, the macro's logic is inserted into the current FBD or LD section. In this case a new section is not created.

Archiving and Documentation

The process for archiving a macro is the same as for archiving and documenting a project (see *Documentation and Archiving, page 729*).

Global / Local Macros

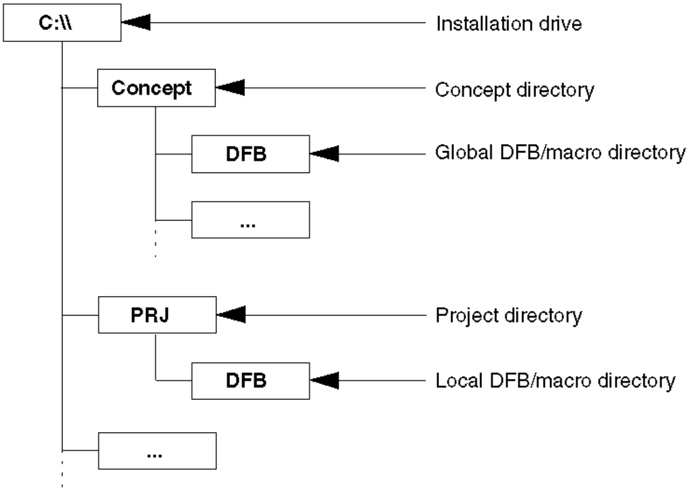
Description

Global and local macros differ in the locality of their directory hierarchy.

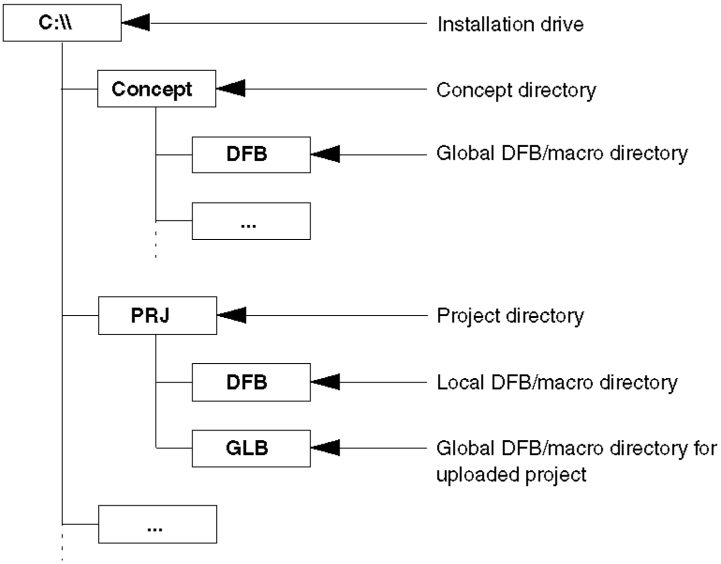
Depending on the directory or subdirectory in which the macro is stored, it can be called up globally, i.e. within all the projects created under Concept, or locally, in a specific project.

In the *Defining the Storage of Global DFBs during Upload, page 1110* you can ensure that during the IEC upload process a GLB directory containing the global macros is produced in the project directory. By doing this, the existing global macros in the **Concept** →**DFB** will not be overwritten and therefore it will not have an effect on other projects.

Directory structure without uploaded project:



Directory structure according to INI settings (**[Upload]: PreserveGlobalDFBs=1**) for uploaded projects:


C:\\
Concept
DFB
...
PRJ
DFB
GLB
...
Installation drive
Concept directory
Global DFB/macro directory
Project directory
Local DFB/macro directory
Global DFB/macro directory for uploaded project

If a local and a global macro have the same name, the name of the local macro is displayed in lower case letters and that of the global macro in upper case letters when they are inserted.

NOTE: The length of the DOS path name in which the macros is stored is limited to 29 characters. Care should be taken that the macro directory does not exceed this limit.

Exchange marking

At a Glance

The exchange markings (@0 to @9) in macros are used to insert the macro in a Concept section. When inserting a macro into a section, you will input a character string that will replace the character strings. It is therefore possible to use a logically identical macro with different variables, data structures and comments, because different series of character strings can be pre-set during each insertion.

The exchange flags can be used in the following elements:
- Section names
- Variable names
- Comments

Comment on exchange markings

A comment on the macro's exchange marking can be written using**File →Section Properties**. This comment will be displayed when the macro is called up in Concept the in the exchange marking's replacement dialog.

Exchange marking in the section name

When a macro is instanced, i.e. when it is called up from an SFC section, a new section is automatically occupied with the name of the macro section, as well as other procedures. The section name must be changed with each instancing so that the macro can be instanced several times in one project. The exchange marking in the section name is used for this. Therefore an exchange marking (@0 to @9) should always be entered when a section is created in the macro. Otherwise the macro can only be called up once from an SFC section and used in the project.

When a macro is called up from an FBD/LD section, the section name of the macro is not significant because no new section is created in this case.

Exchange marking in variable names

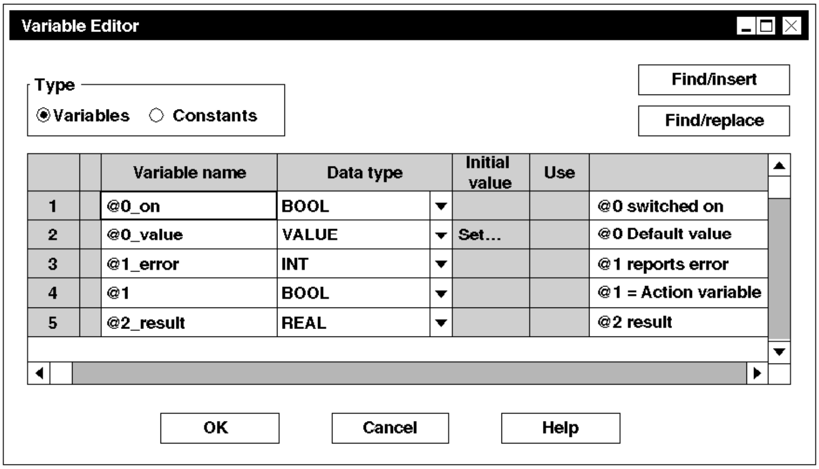
Input and output variables are required to transfer values to or from a network. These variables are already declared in the macro and are connected to the macro's EFBs.

To declare these variables, the variable name (with exchange markings), the data type and possibly a comment (possibly with exchange markings) should be declared in the variables editor. An initial value can also be defined for input variables.

When a macro is instanced in Concept, the exchange markings in all the variable names are replaced with the pre-set character strings. This ensures that the variables required for each use of the macro are clearly declared. If a variable is used in all cases of macro instancing, it should be given a name without the exchange marking.

The same applies to variables with Derived Data Types (data structures). This means that the type of one data structure can be used in as many macros as required as often as required.

Exchange markings in the Variables Editor



Type
● Variables ○ Constants

		Variable name	Data type		Initial value	Use	
1		@0_on	BOOL	▼			@0 switched on
2		@0_value	VALUE	▼	Set...		@0 Default value
3		@1_error	INT	▼			@1 reports error
4		@1	BOOL	▼			@1 = Action variable
5		@2_result	REAL	▼			@2 result

NOTE: If the macro is to be connected as an action to a step in a sequence, it is advisable to denote the variable designated as an action variable only with the @0 exchange marking. In this case, the designated action variable will automatically be connected to the step when the macro is instanced. Care should be taken that the action variables are always of the BOOL type. If the macro contains several action variables (e.g. for the forward and backward running of a motor), it is advisable to define these action variables in a Derived Data Type (data structure) and to denote the variable which this data type is assigned to with the @0 exchange marking only.

Since a clear variable is assigned to each input/output during the instancing of the macro, only unlocated variables can be assigned to the macro when it is created. It is not possible to use direct addresses and located variables in the macro. If located variables are to be used, the corresponding variables can be assigned a direct address in the variables editor after the macro is instanced. If direct addresses are to be used, no variables should be assigned to the corresponding inputs/outputs in the macro and the inputs/outputs should be linked to the address desired after the macro is instanced. If variables have already been declared, they are used (references and initial values are retained).

Exchange marking in comments

When a macro is instanced in Concept, the exchange markings in all the comments are replaced with the pre-set character strings. The same applies to text objects in the section and to variable comments in the variables editor.

Creating Context Sensitive Help (Online Help) for Macros

Introduction

In Concept, help is provided for each EFB, which can be invoked according to the context (the **Help on Type** command button in the EFB properties dialog). There is of course no corresponding help text in Concept for the macros that you created.

You can, however, create your own help for each macro, that can be invoked in Concept with **Help on Type**.

File Format:

You can create your help in the following file formats:

- **.CHM** (Microsoft Windows compiled HTML help file)
- **.DOC** (Microsoft Word format)
- **.HTM** (Hypertext Markup Language)
- **.HLP** (Microsoft Windows help file (16- or 32-Bit Format))
- **.PDF** (Adobe Portable Document Format
- **.RTF** (Microsoft Rich Text Format)
- **.TXT** (Plain ASCII Text-Format)

Name

The name of the help file must be exactly the same as the name of the macro (e.g. SKOE.EXT)

The only exceptions are standardized macro names (e.g. SKOE_BOOL, SKOE_REAL etc.). In these cases the help file name is the macro name without the datatype extension (e.g. macro name) SKOE_BOOL has the help file SKOE.EXT).

Directory

The help file can be stored in the following directories:

- Concept directory
- Concept Help directory (if defined in the file CONCEPT.INI, see readme)
- Global macro directory
- Local macro directory

Invoking the Help File

Concept carries out the following procedure to invoke the help file:

Phase	Description
1	Search for the help file **MacroName.EXT** in the local macro-directory. The help file is searched for in the following sequence: ● .HLP ● .CHM ● .HTM ● .RTF ● .DOC ● .TXT ● .PDF **Result:** If the search result is positive the help file will be displayed, otherwise it will continue with phase 2.
2	Search for the help file **MacroName.EXT** in the global macro-directory. For the order, see phase 1. **Result:** If the search result is positive the help file will be displayed, otherwise it will continue with phase 3.
3	Search for the help file **MacroName.EXT** in the Concept-directory or Concept-Help directory. For the order, see phase 1. **Result:** If the search result is positive the help file will be displayed, otherwise it will continue with phase 4.
4	Display of the comment created in Concept DFB with **Project** →**Properties**.

14.2 Programming and calling up a macro

Overview

This section describes programming and calling up a macro.

What's in this Section?

This section contains the following topics:

Topic	Page
At a Glance	532
Occupying the macro	533
Creating the logic	534
Calling up a macro from an SFC section	537
Calling a macro from an FBD/LD section.	540

At a Glance

At a Glance

Programming and calling up a macro is divided into 3 main steps:

Step	Action
1	Occupying the macro *(see page 533)*
2	Creating the logic *(see page 534)*
3	Calling up the macro in: ● Sequence language (SFC) *(see page 537)* ● Function Block language (FBD) *(see page 540)* ● Ladder Diagram language (LD) *(see page 540)*

Occupying the macro

Description

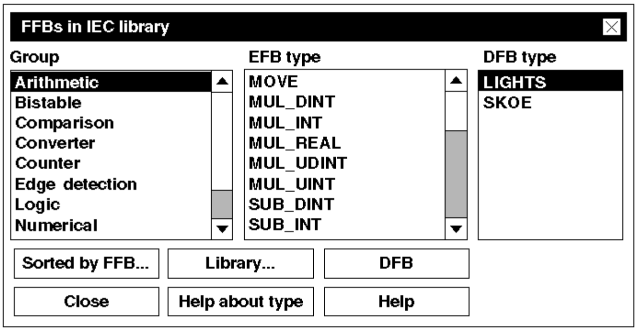
The procedure for occupying the macro is as follows:

Step	Action
1	Close Concept and start Concept DFB.
2	Create a new macro using**File** →**New macro...** menu command. **Reaction:** The name now appears on the title bar: **[untitled]**.
3	Using the menu command **File** →**New section...** generate a new section and enter a section name (with an exchange marking such as @0). The section name (max. 32 characters) must be clear throughout the macro, and it is not case-sensitive. If the section name entered already exists, a warning is given and another name must be chosen. The section name must correspond to the IEC Name conventions, otherwise an error message appears. **Note:** In accordance with IEC1131-3, only letters are permitted as the first character of names. If, however numbers are required as the first character, this can be enabled using the menu command**Presettings** →**Presettings** →**IEC Expansions...** →**IEC Expansions** →**Enable leading figures in identifiers** .
4	Select a programming language for the section: ● Function Block language (FBD) ● Ladder Diagram (LD)
5	The menu command **Project** →**Properties** can be used to generate a comment on the macro. **Reaction:** The comment can then be displayed in Concept using the **Help for type** command key in the selection dialog for macros.
6	The menu command **File** →**Section properties** can be used to generate a comment on the exchange markings. **Reaction:** This comment then appears automatically in the Replace dialog for the exchange markings.
7	Save the macro with the menu command **File** →**Save macro**. **Reaction:** The first time the Save is used, the **Save as** dialog box opens – specify the macro name and directory where it is to be saved here.
8	Select the directory to be occupied by the macro. Attention should be paid to the difference between global and local macros (see also *Global / Local Macros, page 525*).
9	Enter the macro name (max. 8 characters, always with the Extension Mac). The name must be clear throughout the directory, and it is not case-sensitive. If the section name entered already exists, a warning is given and another name must be chosen.

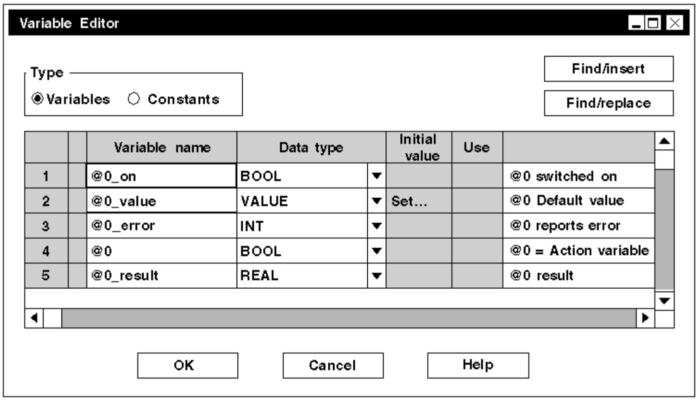
Creating the logic

Description

The procedure for creating the logic is as follows:

Step	Action
1	To insert an FFB into the section, select the menu command **Objects** →**Select FFB...**. **Reaction:** The **FFBs in IEC library** dialog box opens.  FFBs in IEC library Group Arithmetic Bistable Comparison Converter Counter Edge detection Logic Numerical EFB type MOVE MUL_DINT MUL_INT MUL_REAL MUL_UDINT MUL_UINT SUB_DINT SUB_INT DFB type LIGHTS SKOE Sorted by FFB... Library... DFB Close Help about type Help
2	In this dialog box a library can be selected and an FFB selected from it by using the **Library...** command button. Also with the command button **DFB** the manually generated DFBs can be shown and one selected from them.
3	Place the selected FFB in the section.
4	When all FFBs have been positioned, close the dialog box with **Close**
5	Activate the selection mode with **Objects** →**Selection mode**, click on the FFB and move the FFBs to the position required.
6	Activate the link mode with **Objects** →**Link** and connect the FFBs.

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>7</td><td>Activate the Variables Editor with<b>Project</b> →<b>Variables Editor</b> to declare the variables.
For unlocated variables, declare a name here (with exchange markings), a data type, an initial value and a comment if necessary (possibly with exchange markings).
For constants, declare a name here (with exchange markings), a data type, a value and a comment if necessary (possibly with exchange markings).
<b>For example:</b>



Variable Editor

Type ● Variables ○ Constants

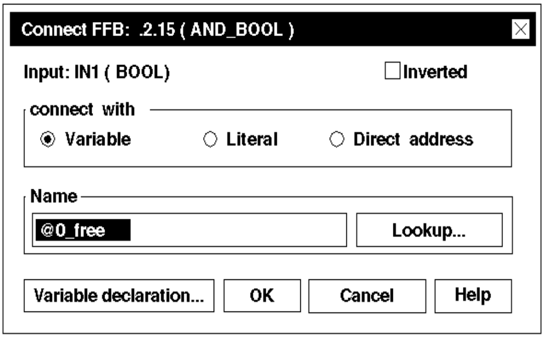
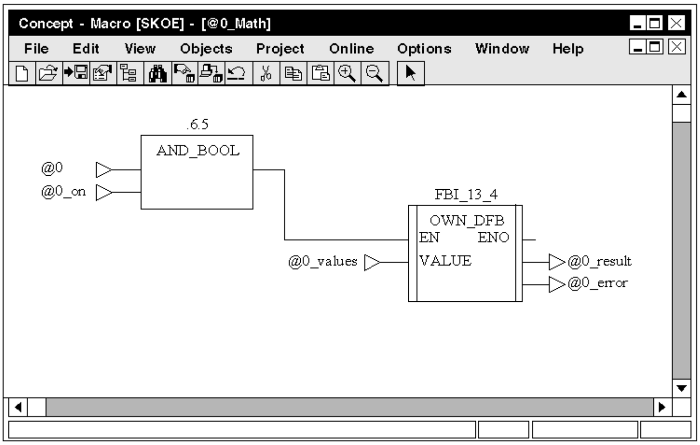
Find/insert

Find/replace

<table>
<tr><th></th><th></th><th>Variable name</th><th>Data type</th><th>Initial value</th><th>Use</th><th></th></tr>
<tr><td>1</td><td></td><td>@0_on</td><td>BOOL ▼</td><td></td><td></td><td>@0 switched on</td></tr>
<tr><td>2</td><td></td><td>@0_value</td><td>VALUE ▼</td><td>Set...</td><td></td><td>@0 Default value</td></tr>
<tr><td>3</td><td></td><td>@0_error</td><td>INT ▼</td><td></td><td></td><td>@0 reports error</td></tr>
<tr><td>4</td><td></td><td>@0</td><td>BOOL ▼</td><td></td><td></td><td>@0 = Action variable</td></tr>
<tr><td>5</td><td></td><td>@0_result</td><td>REAL ▼</td><td></td><td></td><td>@0 result</td></tr>
</table>

OK Cancel Help

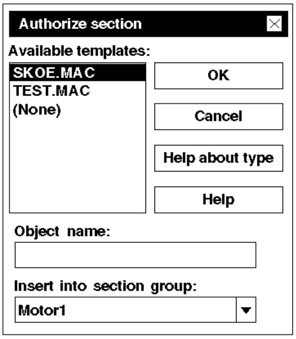
<b>Note:</b> If located variables are to be used, the corresponding unlocated variables can be assigned a direct address in the variables editor after the macro is instanced.
If direct addresses are to be used, no variables should be assigned to the corresponding inputs/outputs in the macro and the inputs/outputs should be linked to the address required after the macro is instanced.
<b>Note:</b> If a variable or constant is to be used in all cases of macro instancing, this variable or constant should be given a name without any exchange marking.</td></tr>
</table>

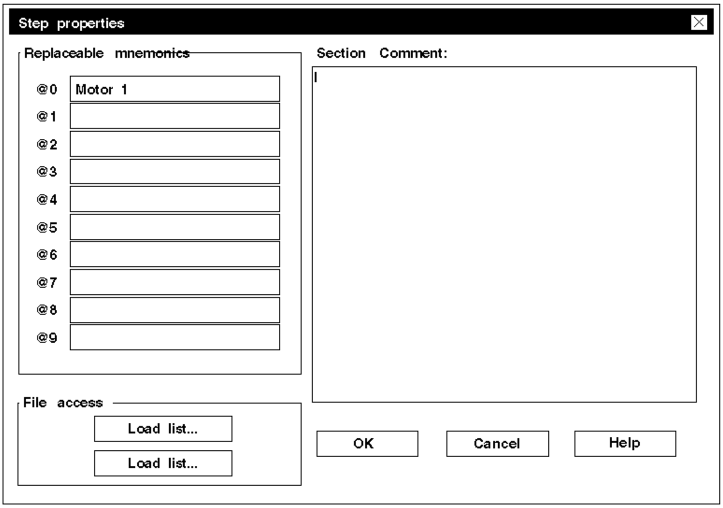
Step	Action
8	Then re-activate the selection mode with **Objects** →**Select** and double-click on one of the unconnected inputs/outputs. **Reaction:** The **Link FFB** dialog box opens, where an actual parameter can be assigned to the input/output.  Connect FFB: .2.15 (AND_BOOL) Input: IN1 (BOOL) □ Inverted connect with ● Variable ○ Literal ○ Direct address Name @0_free Lookup... Variable declaration... OK Cancel Help
9	Save the macro with the menu command **File** →**Save**. **For example:**  Concept - Macro [SKOE] - [@0_Math] File Edit View Objects Project Online Options Window Help .6.5 AND_BOOL @0 @0_on FBI_13_4 OWN_DFB EN ENO VALUE @0_values @0_result @0_error

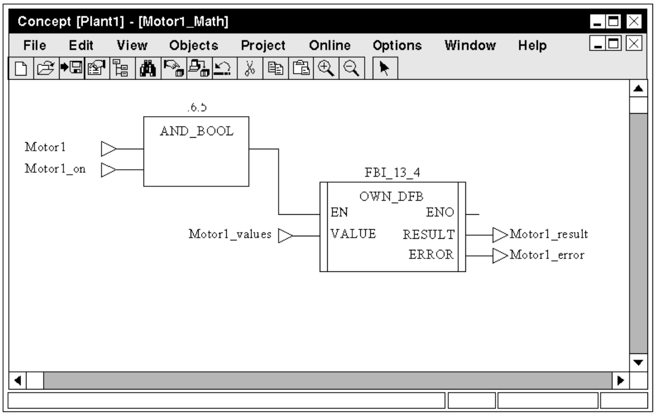
Calling up a macro from an SFC section

Description of the action

The procedure for calling up a macro from an SFC section is as follows:

Step	Action
1	Close Concept DFB.
2	Start Concept, open or create a project and open or create an SFC section.
3	Double-click to open the step properties of the step which the macro is to be connected to.
4	Use the command button **Instance section...** to call up the dialog for instancing the macros.
5	Select the desired macro from the list. If section groups have been created in the Project Browser, the section group where the section is to be inserted can be selected in the **Insert into section group** text field. Confirm with **OK**. Example:  Authorize section Available templates: SKOE.MAC TEST.MAC (None) Object name: Insert into section group: Motor1 **Reaction:** The dialog **Replace** is opened to replace the exchange markings.

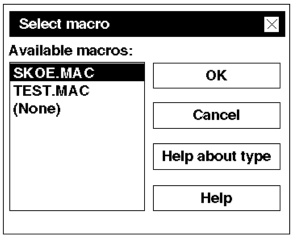
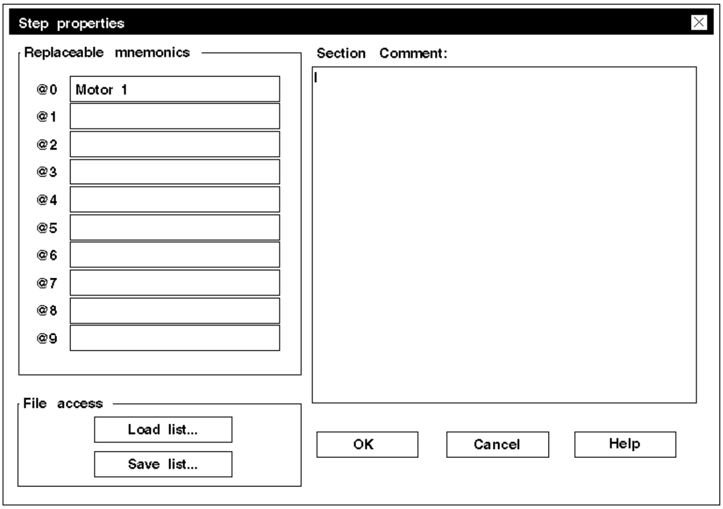
Step	Action
6	Pre-set for the text fields **@0** to **@9** the character strings which the exchange markings are to be replaced with in the macro. Example:  @0 Motor 1 Step properties Replaceable mnemonics @1 @2 @3 @4 @5 @6 @7 @8 @9 Section Comment: File access Load list... Load list... OK Cancel Help

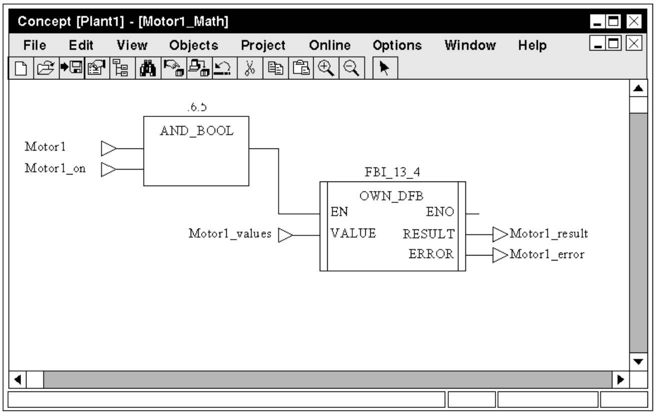
Step	Action
7	Confirm the inputs with **OK**. **Reaction:** The following occurs after the procedure described above has been performed: ● A section is now automatically created whose name consists of the macro section name and of the pre-set character strings in place of the exchange marking. **Note:** This section is not automatically opened. To perform any editing, open by clicking on the variable name in the step properties dialog. ● All the variables declared in the macro are transferred into the variables declaration of the current project and the exchange marking is also replaced with the current character string. If variables have already been declared, they are used (references and initial values are retained). The same applies to any comments containing the exchange flags. ● If the macro contains a single Boolean input variable, it is automatically transferred as an action variable. ● If the macro contains several Boolean input variables, the **Select one of these variables** dialog opens, where the variable desired can be selected as an action variable. ● If a data structure has been marked individually with the exchange flag, the **Select Bool type elements** dialog is called up and the Boolean variable desired for the action can be selected there.
8	This action can be used to call the macro as often as required without any name collisions occurring. The instanced macro and its variables are completely identical to the sections and variables generated beforehand. Example of an instanced macro:  Concept [Plant1] - [Motor1_Math] File Edit View Objects Project Online Options Window Help .6.5 AND_BOOL Motor1 Motor1_on FBI_13_4 OWN_DFB EN ENO VALUE RESULT ERROR Motor1_values Motor1_result Motor1_error

Calling a macro from an FBD/LD section.

Description of the action

The procedure for calling up a macro from an FBD/LD section is as follows:

Step	Action
1	Close Concept DFB.
2	Start Concept, open or create a project and open or create an FBD/LD section.
3	With the menu command **Objects** →**Macro insert** the dialog **Select macro** to insert macros into FBD/LD sections.  Select macro Available macros: SKOE.MAC TEST.MAC (None)
4	Select the desired macro from the list and confirm with **OK**. **Reaction:** The dialog **Replace** is opened to replace the exchange markings.
5	Pre-set for the text fields **@0** to **@9** the character strings which the exchange markings are to be replaced with in the macro. Example:  Step properties Replaceable mnemonics @0 Motor 1 @1 @2 @3 @4 @5 @6 @7 @8 @9 Section Comment: File access

Step	Action
6	Confirm the inputs with **OK**. **Reaction:** The following occurs after the procedure described above has been performed: ● There is now an automatic shift to Insert mode and the macro’s logic can be inserted in any position in the FBD or LD section. ● Moreover, all the variables declared in the macro are transferred into the variable declaration of the current project and the exchange marking is also replaced with the current character string. The same applies to any comments containing the exchange markings.
7	This action can be used to call the macro as often as required without any name collisions occurring. The inserted macro and its variables are completely identical to the sections and variables generated conventionally. Example of an instanced macro:  Concept [Plant1] - [Motor1_Math] File Edit View Objects Project Online Options Window Help .6.5 AND_BOOL Motor1 Motor1_on FBI_13_4 OWN_DFB EN ENO VALUE RESULT ERROR Motor1_values Motor1_result Motor1_error

Variables editor

15

Overview

This Section contains information about declaring variables in the variables editor.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
General	544
Declare variables	545
Searching and replacing variable names and addresses	548
Searching and Pasting Variable Names and Addresses	552
Exporting located variables	556

General

At a Glance

The Variables-Declaration serves as data exchange in user program. Hence you can address Variables (Located and Unlocated Variables) and/or assign a value to constants

Variables or direct addresses will be assigned via the addressing of the I/O-Map and can be used with symbolic names (variable) or with the direct addresses in the programming. In so doing, values will be exchanged between different Sections via the variables or the direct addresses.

NOTE: In accordance with IEC1131-3, only letters are permitted as the first character of variable names. If, however numbers are required as the first character, this can be enabled using the menu command**Options** →**Presettings** →**IEC Expansions...** →**IEC Expansions** →**Enable leading figures in identifiers** enable.

NOTE: Undeclared variables will be denied during programming.

Declare variables

At a Glance

At variable declaration the Data type, address and symbolic name are determined. Via the addressing define the inputs (1x/3x) and outputs (0x/4x), assigned in the user program with the selection of the data type of the respective function, or the respective Function Blocks.

An initial value may also be provided for each variable; this will be transferred into the PLC during the first load.

A comment may be written for each Variable or direct address, to aid recognition of the assignment of a function.

If Declarations are changed, deleted or added, this alteration will be identified through certain symbols in the first column.

Changes in ONLINE mode

Variable names and addresses can be changed online. Apart from that, an unlocated variable can be changed into a located variable (i.e. it will be assigned its own address or the address will be deleted). Clicking on the command button **OK** transfers the changes to the affected sections i.e. the sections in which the changed variables will be used.

This has the following effects:

If…	**Then…**
the variables are modified,	the status of all affected sections will be set to MODIFIED and the affected sections must be loaded into the PLC using **Online →Load modifications**.
a transition section is affected by the modifications,	the SFC section assigned to it is also set to the status MODIFIED.
an affected section is animated,	the animation is aborted.
a modified variable is used in the reference data editor,	no more variables can be inserted into the editor window, and the animation of the reference data editor is stopped. This is valid until the modifications are loaded into the PLC using **Online →Load modifications** and the status EQUAL is restored.

NOTE: The assignment of direct addresses and comments can also occur outside Concept on completion of the programming.

Variable declaration outside the variable editor

Procedure for completing variable declaration outside the variable editor:

Step	Action
1	Export the variable declaration using **File** →**Export** →**Variables: Text delimited**.
2	Open the exported file.
3	Enter the addresses and comments.
4	Import the edited variable declaration using **File** →**Import** →**Variables: Text delimited**.

Copying rows in the variable editor

It is possible to copy individual rows and whole blocks of rows and to paste them into another position in the variable editor, before editing them. This operation is performed using shortcut keys.

Copying and pasting can only take place inside the open variable editor; pasted rows are marked red. These rows must subsequently be changed or they will disappear on exiting the dialog. Identical settings are not permitted in the variable editor.

NOTE: A maximum of 500 rows can be copied.

Procedure for copying and pasting

To copy and paste entire rows proceed as follows:

Step	Action
1	Select the relevant row in the first column in the table. **Reaction:** The entire row is displayed in a different color. **Note:** When copying a block of rows, select the first row in the block, and press **Shift**, while simultaneously selecting the last row in the block.
2	To copy use the shortcut **Ctrl**+**Insert** or **Ctrl**+**Alt**+**c**. **Reaction:** The selected rows are copied into the cache.
3	Select the row off which is to be pasted. **Reaction:** The entire row is displayed in a different color.
4	To paste use the shortcut **Shift**+**Insert** or **Ctrl**+**Alt**+**v**. **Reaction:** The copied rows are pasted off the selected row in the table, and are marked red. **Note:** When pasting between two existing rows, the selected row is moved down according to the number of copied rows.

Printing the variable list

Printing the variable list is done in the main menu **File**. Using the menu command **Print...** open the dialog **Document contents**, in which the print undertaking is set by checking the box **Variable list**.

NOTE: It should be noted that all 32 characters (maximum) of the symbol name do not always appear on the paper when printing.

Searching and replacing variable names and addresses

At a Glance

Use command button **Search/Replace** to call up a dialog box to search and replace variable names and addresses. Therefore, unlike **Search/Insert** the existing variable names/addresses are changed.

Use option button **Name** and **Address** to choose whether to search for variable names or addresses.

If Search and Replace are to be restricted to a certain area of variables or addresses, this area can be selected. In this case, searching and replacing is only carried out in the selected area. If nothing is selected, search and replace are applicable to all variables and addresses in the variable editor.

On activating check box **Extend address** the addresses specified in text box **Address** are automatically extended to Standard format.

Use of wildcards

The following wildcards can be used for searching and replacing:

***** This character is used to represent any number of characters. ***** can only be used at the beginning or the end of a line.

? This character is used to represent exactly one character. If several characters are to be ignored, a certain number of **?** have to be used.

The wildcards can be combined. The combinations ***?** and **?*** are, however, not permitted.

NOTE: When searching and replacing, the number of wildcards in the Search character sequence and the Replace character sequence have to be equal. See also the following examples in the table.

Examples of Search/Replace

The example shows different search methods and the respective results when replacing:

Search:	Replace with:	available names	Result
Name1	Name2	Name1 Name1A Name A Name B	Name2 Name1A NameA NameB
???123	???456	abc123 cde123 abcd123 abc1234	abc456 cde456 abcd123 abc1234
Name1*	Name2*	Name1A Name1B NameAB	Name2A Name2B NameAB
*123	*456	abc123 cde123 abc1234 abcde123	abc456 cde456 abc4564 abcde456
123	*456*	abc123abc cde123defghi abcde123def	abc456abc cde456defghi abcde456def
???123*	???456*	abc123abc cde123defghi abcde123def	abc456abc cde456defghi abcde123def

Search and replace name

Select this option button to search and replace variable names. However, the search for the occurrence of the character sequence to be found is exclusively carried out in column **Variable name** of the variable editor.

Search and replace address

Select this option button, to search and replace addresses. However the search for the occurrence of the address to be found is exclusively carried out in column **Address** of the variable editor.

Search for:

Enter a character sequence, according to which the variables or addresses are to be searched.

Without entering a character sequence that leads to a successful search result, none of the possible functions of the dialog are executed.

NOTE: Entries in the field **Search** remain intact for future use, even after closing the dialog box.

Replace with:

Enter a character sequence, which replaces the character sequence to be searched for in the new variables or addresses

NOTE: Entries in the field **Replace with** remain intact for future use even after closing the dialog box.

Find Next

Description of function **Find Next**:

Stage	Description
1	The command button **Find Next** starts the search process at the beginning of the variable editor table or the selected area and marks the found variable.
2	A query appears, asking whether a search for further occurrences of the character sequence is required.
3	By activating command button **Yes**, the next location of the searched character sequence is selected. By activating command button **No**, the search is terminated.
4	When the search process has reached the end of the variable editor table, the system asks whether or not the search process should be restarted at the beginning of the variable editor table or the selected area.
5	By activating command button **Yes**, the next location of the searched character sequence is selected. By activating command button **No**, the search is terminated.
6	If no further occurrences of the character sequence are found, a message appears, indicating that the search is terminated.

Replace

Description of function **Replace**:

Stage	Description
1	The command button **Replace** starts the search process at the beginning of the variable editor table or the selected area and marks the found variable. **Note:** This function cannot be undone.
2	The system asks whether the found character sequence is to be replaced.
3	By activating command button **Yes**, the variable/address is replaced by the character sequence in the text box **Replace with:** By activating command button **No**, the search is terminated.
4	If there are several uses of the searched character sequence, the next site where it is found is selected and a new query appears.
5	When the search process has reached the end of the variable editor table, the system asks whether or not the search process should be restarted at the beginning of the variable editor table or the selected area.
6	By activating command button **Yes**, the next location of the searched character sequence is selected. By activating command button **No**, the search is terminated.
7	If no further occurrences of the character sequence are found, a message appears, indicating that the search is terminated.

Replace all

Searches for all occurrences of the character sequence and replaces these (without first querying) with the inputs in the text box **Replace with:**. When the search process has reached the end of the variable editor table, the system asks whether or not the search process should be restarted at the beginning of the variable editor table or the selected area.

NOTE: This function cannot be undone.

Searching and Pasting Variable Names and Addresses

Introduction

The **Search/Paste** command button can be used to invoke a dialog for creating new variables based on existing ones. Unlike with Search/Replace, a copy of the existing variables with a new name/address is generated.

If, for example, you have already declared the variables for a motor and you want to declare the same variables but with different names and addresses for another motor, this is easily achieved with this dialog.

If you simply want to generate further variables from a specific range of variables, this area can be selected. In this case, a search will only be carried out in the selected range. If nothing is selected, search and paste applies to all variables in the variable editor.

If you check the Extend Address check box, the addresses entered in the Address text box are automatically extended to Standard format.

Using Wildcards

The following wildcards can be used for searching and pasting:

***** This character is used to represent any number of characters. ***** can only be used at the beginning or the end of a line.

? This character is used to represent exactly one character. If several characters are to be ignored, the corresponding number of **?** have to be used.

The wild cards can be combined. The combinations ***?** and **?*** are, however, not permitted.

NOTE: When searching and pasting, the number of wildcards in the Search string and the Replace string has to be equal.

Find Name

If you select this option button, you can search for variable names. Occurrences of the string to be found are searched for exclusively in the **Variable Name** column of the variable editor.

Find Address

This field is only unavailable for constants.

If you select this option button you can search for addresses. Occurrences of the address to be found are searched for exclusively in the **Address** column of the variable editor.

Find What:

Enter a string to be searched for in variables or addresses.

The search is only carried out in the Variable Name and Address columns in the variable editor table. A search in other areas (e.g. Data type) is not possible.

If you do not enter a string that leads to a successful search result, none of the possible functions of the dialog are executed.

NOTE: Entries in the **Search** field are retained for future use, even after the dialog box is closed.

Replace With:

Enter a string to be replaced in the new variable or address with the string being searched for.

If the name entered already exists, no new variable is created.

NOTE: Entries in the **Replace With** field are retained for future use even after the dialog box is closed.

Offset Address By:

This field is only unavailable for constants.

Enter a value by which the addresses of the existing variables are to be increased.

NOTE: If you do not enter an offset value, the new variable will be placed in the same address as the one already present.

With unlocated variables, it is not necessary to enter a value.

Entries in this field are retained for future use even after the dialog has been closed.

Example of Offset Address By

`SKOE1` has the address `000012`

`Find What: SKOE1`

`Replace With: SKOE2`

`Offset Address By: 1`

This results in the creation of the following new variable:

`SKOE2` on address `000013`

Find Next

Description of **Find Next** function:

Stage	Description
1	The **Find Next** command button starts the search process at the beginning of the variable editor table or the selected area and marks the found variable.
2	A query appears, asking whether a search for further occurrences of the string is required.
3	If the **Yes** command button is pressed, the next location of the string being searched for is marked. If the **No** command button is pressed, the search is finished.
4	When the search process has reached the end of the variable editor table, a query appears asking whether or not the search process should be restarted at the beginning of the variable editor table or the selected area.
5	If the **Yes** command button is pressed, the next location of the string being searched for is marked. If the **No** command button is pressed, the search is finished.
6	If no further occurrence of the string is found, a message appears to inform you that the search is done.

Start Paste

Description of **Start Paste** function:

Stage	Description
1	The **Start Paste** command button is used to start the search process at the beginning of the variable editor table or the selected area and the found variable is marked. **Note:** This function cannot be undone.
2	A query appears asking whether a new variable with the displayed name and address should be created.
3	If the **Yes** command button is pressed, the variable is created and the process continued until all occurrences of the string being searched for have been "exhausted". If the **No** command button is pressed, the search is finished.
4	When the search process has reached the end of the variable editor table, the system asks whether the search process should be restarted at the beginning of the variable editor table or the selected area.
5	If the **Yes** command button is pressed, the next location of the string being searched for is marked. If the **No** command button is pressed, the search is finished.
6	If no further occurrence of the string are found, a message appears to inform you that the search is finished.

Paste All

Searches for all occurrences of the string to be found and replaces them (without asking first) with the new variables given in the **Replace With:** text box. This process is carried out until all occurrences of the string being searched for have been exhausted, or until an error appears.

If an error appears, the function is immediately cancelled. However, all the previously created variables are retained.

NOTE: This function cannot be undone.

Exporting located variables

At a Glance

For data exchange with MMI units, all Located variables in the column **Exp** can be selected and transferred using the Export function in the main menu **File**.

Located variables can be exported via ModLink, Factory Link and via export format "text delimited".

Removing the selection

After export, the selection (in the column **Exp**) of the exported variables using the shortcut **Ctrl**+**Alt**+**F3** can be removed at once.

NOTE: This removal cannot be undone, not even with the command button **Cancel**.

Project Browser

16

Overview

This chapter describes the Project Browser.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
General information about the Project Browser	558
Detailed view in the project browser	561
Operating the Project Browser	563

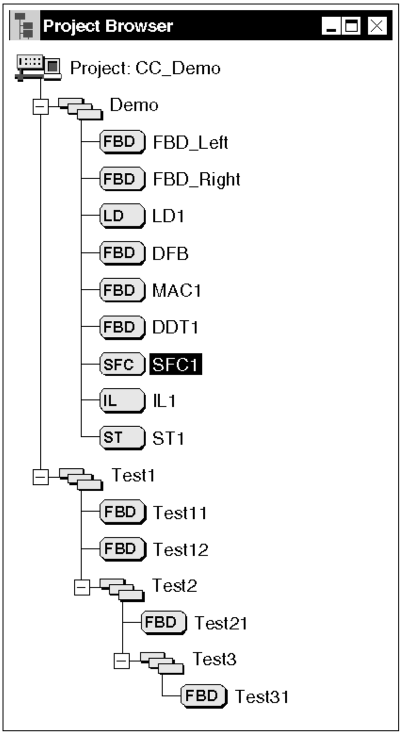
General information about the Project Browser

Introduction

The Project browser can be used to create groups of sections to make the layout clearer and to facilitate operations. These groups have unique names and can contain sections and further section groups. The display and operations are performed graphically by means of Structure tree. The Project browser functions represent a convenient, more extensive way of operating as an alternative to the Concept functions present.

You can open an additional window in the Project browser for viewing existing DFBs, sections with control blocks and transition sections.

Project browser:



Functions

The Project browser provides the following functions:

- Create new section
- Open section (override the editor)
- Changing section properties (names, comments)
- Changing the execution order
- Delete section
- Creating section groups
- Opening section groups (showing the substructure)
- Closing section groups (hiding the substructure)
- Renaming section groups
- Finding section groups or sections in the Project browser
- Moving sections groups or sections (modification of the execution sequence results!)
- Start up offline memory prognosis
- Deleting section groups
- Opening the Configurator
- Minimize open windows
- Open minimized windows
- Close all windows
- Set maximizing window size
- Show exact view
- Excluding individual sections from the alignment between the primary CPU and standby CPU with Hot Standby systems.
- Animate enable states (animation of the structure tree)
- Switch enable state

Restrictions

Attention should be paid to the following restrictions:

- Section groups can only be created with the Project browser.
- Transition sections are not displayed in the Project browser.
- It is only possible to modify the execution sequence via **Project →Execution order** if no section groups exist in the Project browser. After the first section group has been created, no further modifications can be performed via **Project → Execution order** change.
- It is only possible to change the enable status of a section if the variable belonging to the section (.disable) has not been used.

Special features for LL984

Attention should be paid to the following special features when using LL984:

- If one or several LL984 sections exist, the Project browser automatically generates an LL984 section group.
- LL984 sections cannot be moved.
- No IEC sections can be put into or before the LL984 section groups.

Special features of I/O Events and Timer Events

Please take note of the following special features when using interrupt sections:

- If one or several LL984 sections exist, the Project browser automatically generates an I/O Event or Timer Event section group.
- Interrupt sections cannot be moved.
- No IEC sections can be put into or before the interrupt section group.

Detailed view in the project browser

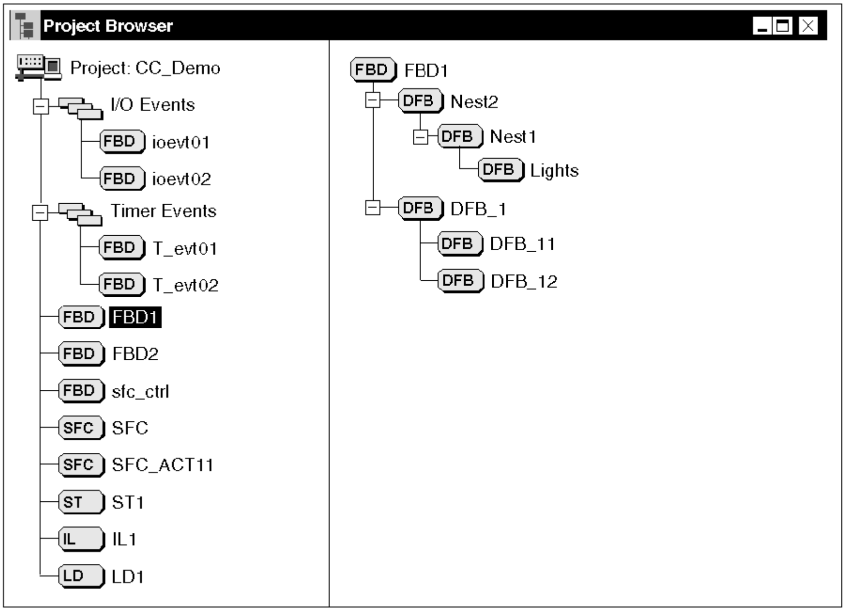
Introduction

In the shortcut menu for the project, you can divide the project browser window vertically using the menu command **Show detailed view**. The right side of the window contains the detailed information concerning the selected element in the project structure tree.

The type of information depends on the selected element:

Element	Information
Project	Call hierarchy for all DFBs used in the project.
Group	No display
LL984 section	No display
FBD/LD	Call hierarchy for all DFBs used in the section. If no DFBs are used, a message is given (!).
ST/IL	Call hierarchy for all DFBs used in the section. If no DFBs are used, or if the analysis fails, a message is given (!).
SFC	The SFC info module can contain the following information: ● Section which contains the control module (e.g. SFC_CTRL) for this SFC section. ● Message with red exclamation point(!): The SFC section is in the execution order before the section with the control module. ● Message with a black exclamation point(!): No transition sections are used. ● All transition sections used.

Detailed view in the right window of the project browser:



Operating the Project Browser

Introduction

The browser allows keyboard and mouse operation.

Mouse operation

Operating the project browser with the mouse:

Function	Key
Selecting a group or section (during selection, a section which is already open is put before all other open sections)	left mouse button
Switching off the context menu	right mouse button
Using the first menu entry of the context menu	Double-click with the left mouse button
Moving a group or section	left-click on the corresponding symbol and hold the mouse button, select the target position by moving the mouse and release the mouse button or **Call context menu (right mouse button) → Select Move →Find target position by cursor up/down →Confirm position with Enter**
Opening or closing a section group	click on the corresponding **+/-** symbol with the left mouse button

NOTE: Context menus do not only appear when symbols are clicked on. The following way to insert a new group or section is available: If the cursor is positioned to the right of the connecting line between two symbols, it changes to show that a context menu can be called in this location by clicking with the right mouse button. This means that a new group or section can be inserted in the line selected.

Keyboard operation

Operating the project browser with the keyboard:

Function	**Key**
selecting the next/previous group/section (during selection, a section which is already open is put before all other open sections)	**Cursor up**/**Cursor down**
Selecting a group/section on the next or previous page	**Scroll up**/**Scroll down**
Selecting a project symbol	**Pos1**
selecting the last group or section	**End**
Scrolling with the keyboard	**CTRL** + **Cursor up**/**Cursor down** or **CTRL** + **Scroll up**/**Scroll down**
Switching off the context menu	**SWITCH** + **F10** or **List**
Carrying out the first menu entry	**Entry**
Moving a group/section	**Call context menu (SWITCH + F10)** → **Select Move** →**Find target position by cursor up/down** →**Confirm position with Enter** or **CTRL + SWITCH** →**Cursor up/down / Scroll up/down** →**Confirm position with Enter**
Opening or closing a section group	**+** or **-** where: **+** restores the status before the last **-**
Opening a section group and all sub-groups	*****
Deleting a group or section	**Delete**
Selecting the group above	**Cursor left** or **backspace delete** If the element actually selected is a group when **cursor left** is used, the group is closed before the higher group is selected.
Selecting the first section/group in a group	**Cursor right** If the group is closed and contains a section or groups, it is opened.
Canceling the move	**ESC**

Derived data types

17

Overview

This Chapter describes the data type editor and the procedure for creating derived data types.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
17.1	General information on Derived Data Types	566
17.2	Syntax of the data type editor	574
17.3	Derived data types using memory	586
17.4	Calling derived data types	588

17.1 General information on Derived Data Types

Overview

This section contains general information about Derived Data Types.

What's in this Section?

This section contains the following topics:

Topic	Page
Derived Data Types	567
Global / Local Derived Data Types	570
Extended Data Type Definition (larger than 64 Kbytes)	572

Derived Data Types

Introduction

Derived data types are defined using the data type editor. All the elementary data types that already exist in a project and the Derived Data Types can be used to define new data types.

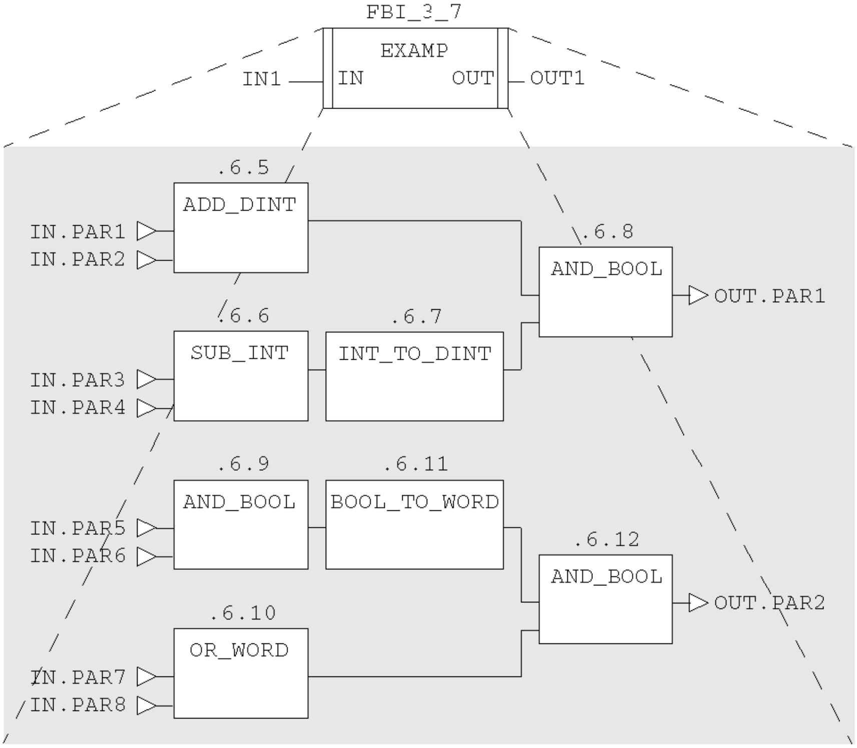
NOTE: Open the Data Type Editor in Concept/Concept-DFB using **File** →**Open** → **File Format Data Type Files (*.DTY)**.

NOTE: Note that the **File** →**Save** and **File** →**Save as** menu commands are not available in this editor. To save the Derived Data Types, select the menu command **File** →**Exit**.

Using Derived Data Types

Various block parameters can be transferred as one set through Derived Data Types. This set is then divided into individual parameters again in the DFBs and EFBs; these are processed and then output again as a set or as individual parameters.

Using Derived Data Types in a DFB:



NOTE: For a definition of the Derived Data Types IN and OUT, see *Example of a Derived Data Type, page 576*.

Definition of Derived Data Types

The definition of Derived Data Types appears in textual form.

When text is entered, all the standard Windows services for word processing are available. The data type editor also contains some further commands for text processing.

Spelling is immediately checked when key words, separators and comments are entered. If a key word, separator or comment is recognized, it is identified with a color surround.

Name Conventions

The following name conventions apply to derived data types:

- **Multi-element variable**
  If a Derived Data Type is assigned to a variable (field or structure), it is designated as a multi-element variable.
- **Structured variable**
  If a derived data type is assigned to a variable consisting of several elements, it is designated as a structured variable. If this is the case, the declaration contains the keyword STRUCT *(see page 577)*. This also applies if the derived data type only contains ARRAY declarations.
  e.g.
  ```
  TYPE
    EXP:
     STRUCT
     PAR1: ARRAY [0..1] OF INT;
     PAR2: REAL;
     PAR3: TEST;
      END_STRUCT;
  END_TYPE
  ```
- **Field variable**
  If a derived data type is assigned to a variable which consists of several ARRAY Declarations *(see page 578)*, it is designated as a field variable. The key word STRUCT is not used in this case.
  e.g.
  ```
  TYPE
    TEST: ARRAY [0..1] OF UINT;
  END_TYPE
  ```

Global / Local Derived Data Types

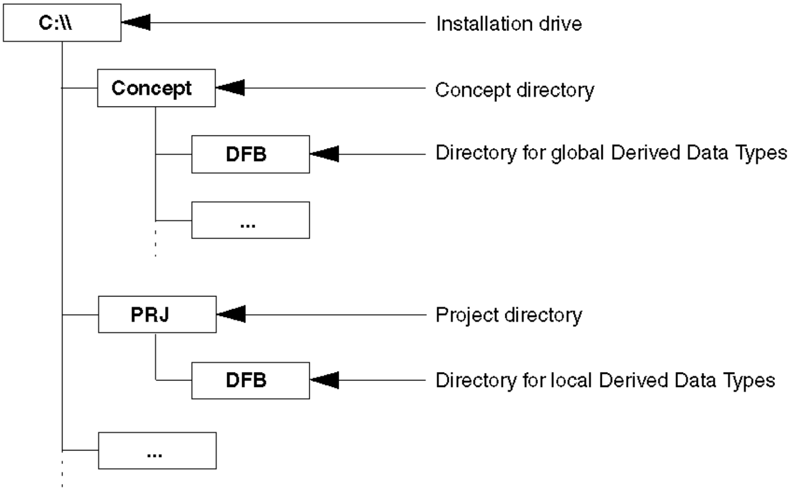
Description

Concept differentiates between global Derived Data Types and local Derived Data Types. Global Derived Data Types can be used in any project (Concept) or in any DFB (Concept DFB). Global Derived Data Types must be placed in the DFB subdirectory of the Concept Directory. Local Derived Data Types are only recognized in the context of a project or its local DFBs and can only be used there. Local Derived Data Types must be located in the DFB subdirectory of the project directory.

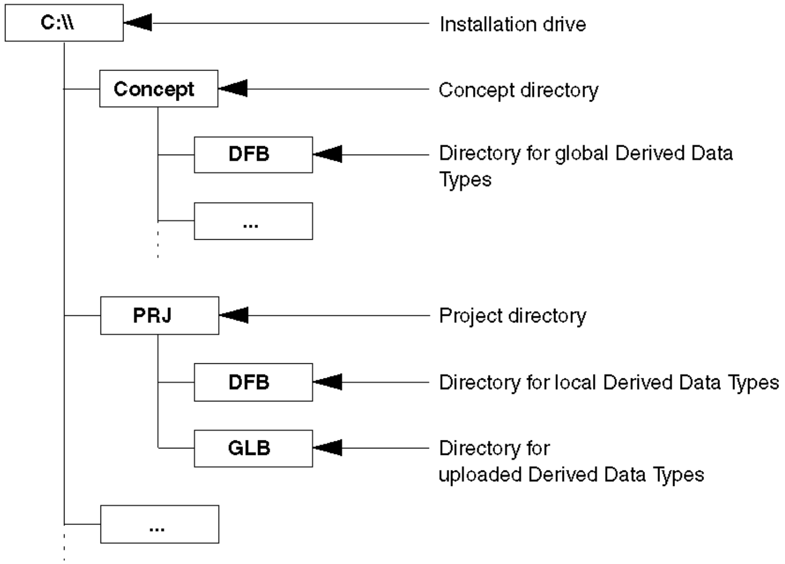
In the *General information on the Concept INI file, page 1107* you can specify that a GLB directory containing the global Derived Data Types is generated in the project directory during the IEC upload process. This means existing global Derived Data Types in **Concept** →**DFB** are not overwritten, and there is no effect on other projects.

NOTE: This file structure should be noted at the creation stage of the Derived Data Types, because the menu command **File** →**Save as** is not available for these. For this reason it is imperative to ensure that the correct path has been selected prior to pressing **OK**.

Directory structure without uploaded project:



Directory structure after setting up INI file (**[Upload]: PreserveGlobalDFBs=1**) for uploaded projects:


C:\\
Installation drive
Concept
Concept directory
DFB
Directory for global Derived Data Types
...
PRJ
Project directory
DFB
Directory for local Derived Data Types
GLB
Directory for uploaded Derived Data Types
...

Number of data type files

Concept only supports one single local data type file for each project and only one single global data type file. To ensure consistency between the host computer and the PLC, the project containing one of the Derived Data Types must be reloaded into the PLC after either of these files is edited.

If a local and a global Derived Data Type have the same name, the local Derived Data Type is given priority.

Maximum File Size

NOTE: The maximum file size (.DTY) for global and local Derived Data Types (i.e. the definitions and including all comments) is 64 kbytes. If this maximum file size is too small, the data type definitions can be shared between the global and local data type file. T

Extended Data Type Definition (larger than 64 Kbytes)

At a Glance

The maximum file size (*.dty) for global and local derived data types is 64 KBytes (this includes the definitions and all comments). To extend this limitation for **local** derived data types, you can create an Include file (*.inc), without increasing the size of the database. This file contains a list of any data type files with the extension *.ddt. However, the file cannot contain any DTY data type files.

A DDT data type file is structured just like a DTY data type file. Unlike DTY data type files, a backup copy is not made in the database for DDT data type files. Therefore it is impossible to determine exactly which data type was recently changed. Each data type in the DDT data type file looks as if it was changed if the DDT data type file was changed in any location. All initial values for variables with data types defined in this DDT data type file are set to 0. The program status will be NOT EQUAL as well.

The Include file is only allowed to be in the local DFB directory and contains the name of the project, e.g. TESTPRJ.INC. Changing an Include file is monitored with check digits.

The Include file has priority over the DTY data type file.

NOTE: Only one Include file can be in the local DFB directory.

The definition of **global** derived data types has not changed.

Create INC file

An Include file can only contain existing data type files (*.ddt), i.e. the data type files must exist in the project before creating an Include file.

The DDT data type files can be compared to DTY data type files, they are created in the same way *(see page 575)* and can therefore have the same contents.

The Include file is created in the Include file editor.

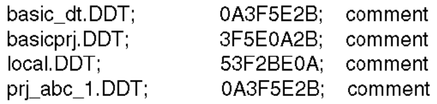
Carry out the following steps to open the Include file editor:

Step	Action
1	Select **File** →**Open** and then go to the **List files of type** list box and select the option **Datatype file (*.dty...)**. **Reaction:** The file types *.dty,*.ddt,*.inc are shown in the **File name** text box.
2	In the **Folder** text box, you must select the local DFB directory for your project.
3	In the **File name** text box, delete all data types except for *.inc.
4	Enter the name of the project as file name, e.g. TESTPRJ.INC.
5	Select **OK** and another window is opened. Confirm the question of if this file should be created with **Yes**. **Reaction:** The Include file editor is opened.

With this editor, the Include file created is automatically opened and now contains all data type files (*.ddt) in the project. The data type files can then be added to the contents of the Include file to define the Include file.

Only file names are allowed for data type file list, no path entries.

Example of the contents of an Include file:



```
basic_dt.DDT;        0A3F5E2B;   comment
basicprj.DDT;        3F5E0A2B;   comment
local.DDT;           53F2BE0A;   comment
prj_abc_1.DDT;       0A3F5E2B;   comment
```

The check digits are automatically generated by Concept when opening the project.

Limitations

Changes in a DDT data type file or in the Include file do not cause this data type check. Concept automatically carries out a data type check. The check consists of many general tests which require a large amount of time.

This smallest change causes the program status to go to NOT EQUAL.

17.2 Syntax of the data type editor

Overview

This section describes the syntax to be noted when generating Derived Data Types.

What's in this Section?

This section contains the following topics:

Topic	Page
Elements of the Derived Data Types	575
Key Words	577
Names of the derived datatypes	582
Separators	583
Comments	585

Elements of the Derived Data Types

At a Glance

The following elements can be used to generate the Derived Data Types:

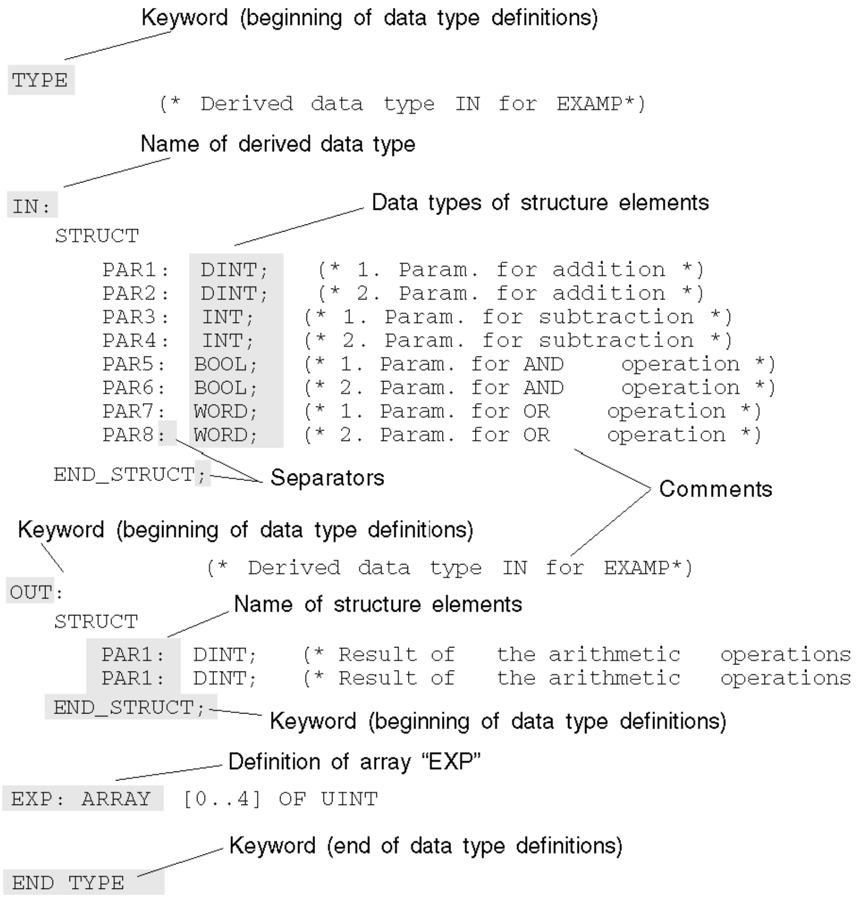
- Key words *(see page 577)*
- Names *(see page 582)*
- Separators *(see page 583)*
- Comments *(see page 585)*

Indents

Indents and line breaks can be inserted at any position where a blank character is also allowed to make the layout clearer. This does not affect the syntax.

Example of a Derived Data Type

Defining Derived Data Types:



```
TYPE
         (* Derived data type IN for EXAMP*)
IN:
   STRUCT
      PAR1:  DINT;   (* 1. Param. for addition *)
      PAR2:  DINT;   (* 2. Param. for addition *)
      PAR3:  INT;   (* 1. Param. for subtraction *)
      PAR4:  INT;   (* 2. Param. for subtraction *)
      PAR5:  BOOL;  (* 1. Param. for AND    operation *)
      PAR6:  BOOL;  (* 2. Param. for AND    operation *)
      PAR7:  WORD;  (* 1. Param. for OR    operation *)
      PAR8:  WORD;  (* 2. Param. for OR    operation *)
   END_STRUCT;

            (* Derived data type IN for EXAMP*)
OUT:
   STRUCT
      PAR1:  DINT;   (* Result of   the arithmetic   operations
      PAR1:  DINT;   (* Result of   the arithmetic   operations
   END_STRUCT;

EXP: ARRAY  [0..4] OF UINT

END TYPE
```

Keyword (beginning of data type definitions)
Name of derived data type
Data types of structure elements
Separators
Comments
Keyword (beginning of data type definitions)
Name of structure elements
Keyword (beginning of data type definitions)
Definition of array "EXP"
Keyword (end of data type definitions)

Key Words

Introduction

The following key words can be used to define the Derived Data Types:

- TYPE ... END_TYPE *(see page 577)*
- STRUCT ... END_STRUCT *(see page 577)*
- ARRAY *(see page 578)*
- "Data types" *(see page 581)*

In accordance with IEC 113-3, key words must be entered in upper case. If lower case is also to be used, however, this can be enabled in the dialog box **IEC Extensions** using the option **Allow case insensitive keywords**.

If a key word is recognized, it is identified in colour.

TYPE ... END_TYPE

The key word TYPE denotes the beginning of the data type definitions. The key word TYPE is only entered once at the beginning of the data type definitions and is then valid for all subsequent data type definitions.

The key word END_TYPE denotes the end of the data type definitions. The key word END_TYPE is only entered once at the end of the data type definitions.

STRUCT ... END_STRUCT

The key word STRUCT denotes the beginning of the elements of a Derived Data Type. Structures are collections of various Elementary and Derived Data Types. Variables, to which a Derived Data Type like this is assigned, are designated as structured variables.

The key word END_STRUCT denotes the end of the elements of a Derived Data Type.

Syntax for STRUCT

```
STRUCT
NAME1: Data type;
NAME2: Data type;
NAMES: Data type;
END_STRUCT;
```

Example: STRUCT ... END_STRUCT

```
TYPE
   Example1:
    STRUCT
      Name1: BOOL; (* Comment *)
      Name2: INT; (* Comment *)
      Name3: ARRAY [0..5] OF BOOL; (* Comment *)
    END_STRUCT;
END_TYPE
```

ARRAY

If several consecutive elements with the same data type are in use, they can be defined as a field with the key word ARRAY.

After the key word ARRAY, the zone is given, i.e. the number of elements and the number of the elements' sub-elements if need be. Finally, the data type common to all the elements is given. Elementary or Derived Data Types can also be used.

If a Derived Data Type is assigned to a variable in the variable editor consisting of an ARRAY declaration, it is designated as a field variable.

Syntax for ARRAY

NAME: ARRAY [No. 1.Element .. No. last element, no. 1st element .. no. last element etc.] OF data type;

Encapsulation Depth

The encapsulation depth is practically unlimited but should be restricted to a few stages, e.g. to 2 or 3 dimensions to ensure clarity. The maximum size of a data type file should not exceed 64KB.

Restrictions

ARRAY indices can not be used in **generic** functions/function blocks (i.e. SEL and MUX).

The following operations would generate an error:

```
k := Arr[a,b,MUX(i,in1=2)];
```

```
Arr30[0,1,MUX_INT( K := K, IN0 := 0, IN1 := 1, IN2 := 0)];
```

ARRAY indices can be used in all other functions/function blocks.

The following operation is possible:

```
B[8] :=  Arr3[REAL_TO_INT(TAN_REAL(ie.real1[2]),j,2]);
```

Example: One-dimensional ARRAYs

In the following example, a Derived Data Type is defined with the name par. This Derived Data Type contains 6 elements (par[0] to par [5]) of the BOOL data type.

```
par: ARRAY [0..5] OF BOOL;
```

It is not absolutely necessary to begin the range with "0". Any range can be defined. In this example the Derived Data Type contains 14 elements (par[51] to par [64]) of the BOOL data type.

```
par: ARRAY [51..64] OF BOOL;
```

Example: A one-dimensional ARRAY in a structured variable

ARRAYs can also be used as elements in structured variables (definition with the key word STRUCT):

```
Par3: STRUCT
         Name1: ARRAY [0..5] OF INT);
         Name2: BOOL;
         Name3: REAL;
       END_STRUCT;
```

Variables of the Par3 data type contain 3 elements:

- Name1 with 6 sub-elements (Par3.Name1[0] to Par3.Name1[5] of the INT data type
- Name2 with 1 element of the BOOL data type
- Name3 with 1 element of the REAL data type

Multi-dimensional ARRAYs

In multi-dimensional ARRAYs the statements in [ ] are expanded by the number of sub-elements of each element. i.e. the element given in the ARRAY contains in turn a specific number of elements of the same data type.

Example: Two-dimensional ARRAY

The following example shows a two-dimensional ARRAY.

```
Par4: ARRAY [0..5, 1..3] OF BOOL;
```

Variables of the Par4 data type contain 6 elements of the BOOL data type each with 3 sub-elements of the BOOL data type:

- Par4 [0,1] to Par4 [0,3]
- Par4 [1,1] to Par4 [1,3]
  and so on up to
- Par4 [5,1] to Par4 [5,3]

Example: Three-dimensional ARRAY

The following example shows a three-dimensional ARRAY.

```
Par5: ARRAY [0..5, 1..4, 11..14] OF REAL;
```

Variables of the Par5 data type contain 6 elements of the REAL data type each with 4 sub-elements of the REAL data type: Each sub-element contains 4 further sub-elements of the REAL data type:

- Par5 [0,1,11] to Par5 [0,1,14]
- Par5 [0,2,11] to Par5 [0,2,14]
  and so on up to
- Par5 [0,4,11] to Par5 [0,4,14]
- Par5 [1,1,11] to Par5 [1,1,14]
  and so on up to
- Par5 [5,4,11] to Par5 [5,4,14]

Example: A multi-dimensional ARRAY in a structured variable

As for one-dimensional ARRAYs, multi-dimensional ARRAYs can also be used as elements in structured variables (definition with the key word STRUCT).

```
Par6: STRUCT
         Name1: ARRAY [0..5, 1..3] OF INT;
         Name2: BOOL;
         Name3: REAL;
       END_STRUCT;
```

Variables of the Par6 data type contain 3 elements:

- Name1 with 18 sub-elements:
  - Par6.Name1[0,1]
    to
  - Par6.Name1[5,3] of the INT data type
- Name2 with 1 element of the BOOL data type
- Name3 with 1 element of the REAL data type

Example: Step by step definition of multi-dimensional ARRAYs

Multi-dimensional ARRAYs can also be defined step-by-step.

```
Par71: ARRAY [1..100] OF WORD;
Par72: ARRAY [1..3] OF Par71;
Par73: ARRAY [1..33] OF Par6;
```

"Data types"

The names of the elementary data types and the names of already defined Derived Data Types are recognized as a key word (in contrast with the names of elementary data types, the names of derived data types are not displayed in color). Data types must be closed with the separator ";".

If a different Derived Data Type is in use while defining a Derived Data Type, it must be defined before it can be invoked.

Names of the derived datatypes

Description

Names are given to the derived data types and the elements in the data type editor.

Names should not be longer than 24 characters and must be ended with the separator ":"

Names are displayed in black

NOTE: Names should not begin with figures even if the option **Options** → **Preferences** →**IEC expansions...** →**Enable leading figures in identifiers** is activated.

NOTE: Within the data type editor it is possible to use special symbols (umlauts, accents etc.). These symbols are also permitted in Concept EFBs created with Concept-EFB can **NOT** be used however. The above is based on the internal processes of Borland products. It is therefore strongly recommended that **NO** special symbols are used in names.

Separators

Introduction

The following separators can be used to define the derived data types:

- : (colon) *(see page 583)*
- ; (semi-colon) *(see page 583)*
- [ ] (square brackets) *(see page 583)*
- .. (full stops) *(see page 584)*

Separator ":" (colon)

Marks the end of a name (name of the derived data type, name of the element).

Example:

```
TYPE
   Example1:
    STRUCT
      Name1: BOOL; (* Comment *)
      Name2: INT; (* Comment *)
      Name3: ARRAY [0..5] OF BOOL; (* Comment *)
    END_STRUCT;
END_TYPE
```

Separator ";" (semi colon)

Indicates the end of an instruction.

Example:

```
TYPE
   Example1:
    STRUCT
      Name1: BOOL; (* Comment *)
      Name2: INT; (* Comment *)
      Name3: ARRAY [0..5] OF BOOL; (* Comment *)
    END_STRUCT;
END_TYPE
```

Separator "[ ]" (square brackets)

Encloses the range specification of the keyword ARRAY.

Example:

```
TYPE
   Example1:
    STRUCT
      Name1: BOOL; (* Comment *)
      Name2: INT; (* Comment *)
      Name3: ARRAY [0..5] OF BOOL; (* Comment *)
    END_STRUCT;
END_TYPE
```

Separator ".." (full stops)

Separates the beginning and end of range for the keyword ARRAY.

Example:

```
TYPE
   Example1:
    STRUCT
      Name1: BOOL; (* Comment *)
      Name2: INT; (* Comment *)
      Name3: ARRAY [0..5] OF BOOL; (*Comment *)
    END_STRUCT;
END_TYPE
```

Comments

Description

In the data type editor begin comments with the character sequence (* and end with the character sequence *). Between these character sequences any comments can be entered.

Comments can be entered at any position in the data type editor

Comments are displayed in color.

Using the menu command **Options →Preferences →IEC Extensions →Nested comments authorized** you can enable nested comments to be authorized. There are then no limits to the nesting depths.

Example: Comments

```
TYPE
   Example1:
   STRUCT
         Name1: BOOL; (* Comment *)
         Name2: INT; (* Comment *)
         Name3: ARRAY [0..5] OF BOOL; (* Comment *)
       END_STRUCT;
END_TYPE
```

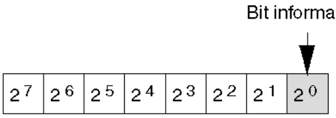
17.3 Derived data types using memory

Use of Memory by Derived Data Types

Boolean Elements

Boolean elements are conveyed as bytes, the bit information is in the first bit.

Storage of Boolean elements:



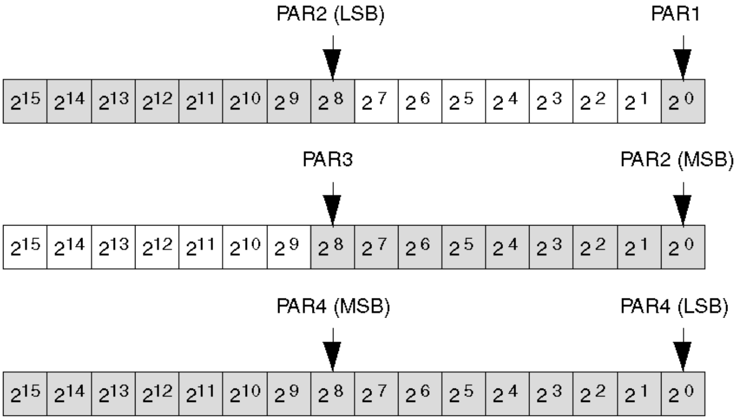
WORD Elements

There are no gaps when Derived Data Types are stored in memory.

Example of a Derived Data Type:

```
TYPE
  SKOE:
   STRUCT
   PAR1: BOOL;
   PAR2: WORD;
   PAR3: BOOL;
   PAR4: WORD;
   END_STRUCT;
END_TYPE
```

Storage of the Derived Data Type in memory:



It should be ensured that WORD elements begin with word addresses (a dummy bit could be inserted).

NOTE: If the structured variable is associated with a direct address and is further processed externally (e.g. is read by a visualisation system from the PLC), the WORD elements (including ANY_NUM elements) absolutely must begin with a word address.

Located Derived Data Types

If derived data types are passed to the hardware (located Derived Data Types) they may only be stored in the 3x or 4x registers. Storage in the 0x or 1x registers is not possible.

17.4 Calling derived data types

Calling Derived Data Types

Introduction

When a derived data type is defined in the data type editor, the name of the derived data type appears automatically in the variables editor (Column **Data type**). The assignment of a variable to a derived data type occurs in the same way as for elementary data types.

Multi-element variables can be called as a text input of the individual elements or using a dialog box **Lookup variables**. In such a case, the corresponding elements are chosen according to the selection of a multi-element variable in the **Select Component of Type** dialog box.

Addressing a structure element

To address a structure element the variable names are first assigned and then separated from the element name by a dot (e.g.VARIABLE_NAME.ELEMENT_NAME). If this element also consists of a Derived data type as well, it is again separated from the next element name by a full stop (e.g. VARIABLE_NAME.ELEMENT_NAME.SUB_ELEMENT_NAME) etc.

Example: Addressing a structure element

Addressing a structure element:

Step	Action

1	Define a derived data type.
	For example:
	`TYPE`
	`   Example1:`
	`   STRUCT`
	`      Par1: BOOL;`
	`      Par2: INT;`
	`   END_STRUCT;`
	`END_TYPE`

2	Declare a new variable in the variable editor (e.g. with the name TEST).
3	Assign these variables the data type of the derived data type created (e.g. Example1).

Step	Action
4	Close the variable editor with **OK**. **Reaction:** A new multi-element variable called "TEST" of data type "Example1" is now created.
5	To address this multi-element variable in its "entirety", simply enter the name of the variable (TEST) into the program as usual. To address only a single element of this multi-element variable (e.g. the element "Par1"), enter the variable name and (separated by a dot) the name of the element (e.g. TEST.Par1) into the program.

Addressing an ARRAY element

To address an ARRAY element the variable name comes first followed by the element number in square brackets (e.g. VARIABLE_NAME[4]).

Example: Addressing an ARRAY element

Addressing an ARRAY element

Step	Action

1	Define a derived data type.
	For example:
	`TYPE`
	`   Example2: ARRAY [0..5] OF BOOL;`
	`END_TYPE`

2	Declare a new variable in the variable editor (e.g. with the name MY_VAR).
3	Assign these variables the data type of the derived data type created (e.g. Example2).
4	Close the variable editor with **OK**. **Reaction:** A new multi-element variable called "MY_VAR" of data type "Example2" was created.
5	To address this "entire" multi-element variable, simply enter the name of the variable (MY_VAR) into the program as usual. To address only a single element of this Multi-element variable (e.g. the 4th element of the ARRAY), enter into the program the variable name and in square brackets the number of the element (e.g. MY_VAR[4]).

Addressing an ARRAY element in a structure

To address an ARRAY element which is part of a structure the variable name is entered first, followed by a dot and the element name, followed by the element number in square brackets (e.g. VARIABLE_NAME.ELEMENT[4])

Example: Addressing an ARRAY element in a structure

Addressing an ARRAY element in a structure:

Step	Action

1	Define two derived data types (in which the second derived data type uses the first as an element).
	For example:
	`TYPE`
	`   Example3:`
	`   STRUCT`
	`      Par1: BOOL;`
	`      Par2: ARRAY [0..5] OF BOOL;`
	`      Par3; BOOL;`
	`   END_STRUCT;`
	`   Example4:`
	`   STRUCT`
	`      Elem1: Example3:`
	`      Elem2: INT;`
	`   END_STRUCT;`
	`END_TYPE`

2	Declare a new variable in the variable editor (e.g. with the name COMPLEX_VAR).
3	Assign these variables the data type of the derived data type created (e.g. Example4).
4	Close the variable editor with **OK**. **Reaction:** A new multi-element variable called "COMPLEX_VAR" of data type "Example4" is now created.
5	To address this "entire" multi-element variable, simply enter the name of the variable (COMPLEX_VAR) into the program as usual. For example, if you only want to address one individual element of this multi-element variable (e.g. you want to call the 5th element of the ARRAY from element "Par2" (derived data type "Example3") as an element of "Elem1"), enter the variable names in your program and the element name separated by a dot, (in your "current" derived data type, here "Example4"), and the name of the elements of the derived data type called by the "current" derived data type separated by a dot (here "Example3") and followed by the element number in square brackets (e.g. COMPLEX_VAR.Elem1.Par2[5]).

Range Monitoring for Indexed Access

Indexed access to Arrays in ST are monitored for over range violations. If the index is a constant, monitoring is carried out on the compile level in the programming device. If the index is a variable, monitoring is carried out during runtime in the PLC during every cycle.

In order to optimize program run time, the index for multi-dimensional arrays or arrays that are embedded in structures are only checked for the starting and end address of the memory area reserved for the variable. This means that an invalid component is overwritten even though it is always located inside the structure. An error message is only generated in the event display dialog box when the index for the memory area allocated for this structure is exited: "ARRAY Index exceeds range (..)". Data access is diverted to the memory starting address of the structure.

⚠ CAUTION

Data can be overwritten!

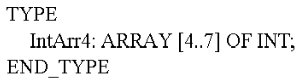
The index ARRAY does not serve as the range boundary, but always the entire memory range allocated to the variable.

With multi-dimensional Arrays or Arrays within a structure, an error message is first returned when the index is displayed on a memory address outside of the memory area allocated for the entire array or entire structure.

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

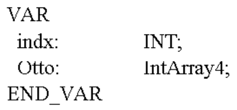
Example 1 one dimensional structure

Defining a derived data type in the data type editor:



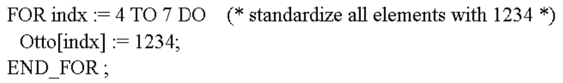
```
TYPE
   IntArr4: ARRAY [4..7] OF INT;
END_TYPE
```

Variable definition:



```
VAR
  indx:          INT;
  Otto:          IntArray4;
END_VAR
```

Sequence in text language:

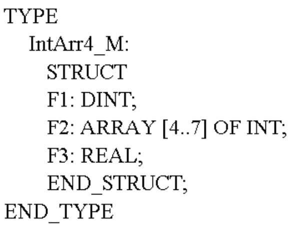


```
FOR indx := 4 TO 7 DO    (* standardize all elements with 1234 *)
  Otto[indx] := 1234;
END_FOR ;
```

If the Index (**indx**) is too large (>7) or too small (<4), and data access is made outside the range(**Otto**), the first element is automatically accessed in the PLC runtime system (**Otto[4]**) and an error message is generated.

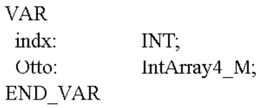
Example 2 Array embedded in a structure

Defining a derived data type in the data type editor:



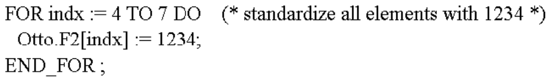
```
TYPE
   IntArr4_M:
      STRUCT
      F1: DINT;
      F2: ARRAY [4..7] OF INT;
      F3: REAL;
      END_STRUCT;
END_TYPE
```

Variable definition:



```
VAR
 indx:          INT;
 Otto:          IntArray4_M;
END_VAR
```

Sequence in text language:



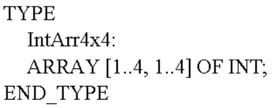
```
FOR indx := 4 TO 7 DO    (* standardize all elements with 1234 *)
  Otto.F2[indx] := 1234;
END_FOR ;
```

In this case the range boundary is determined by the total amount of memory occupied by the **Otto** variables. The range monitoring is first activated when **indx <2** or **indx >9** occurs. An over range then accesses the address **Otto.F1**!

Access with **indx = 2-3** or **indx = 8-9** is not recognized as faulty, but the elements **F1** (indx = 2-3)or **F3** (indx = 8-9) are overwritten!

Example 3 multi-dimensional array

Defining a derived data type in the data type editor:



```
TYPE
   IntArr4x4:
   ARRAY [1..4, 1..4] OF INT;
END_TYPE
```

Variable definition:

```
VAR
  indx_x:       INT;
  indx_y:       INT;
  Otto:         IntArray4x4;
END_VAR
```

Sequence in text language:

```
FOR indx_x :=1 TO 4 DO    (* standardize all elements with 1234 *)
 FOR indx_y :=1 TO 4 DO
  Otto[indx_x, indx_y] := 1234;
 END_FOR ;
END_FOR ;
```

In this case when the first index **indx_x** of the range boundary is exceeded it directly results in a error message. For the second index **indx_y**, the range monitoring becomes active when the address created from the two indexes are outside the memory area for the entire array (4*4 words).

Examples:

for **indx_x = 1**, it can become **indx_y = 16** before the range monitoring is put into effect.

for **indx_x = 4**, range monitoring becomes active when **indx_y = 5**.

Reference data editor

18

Overview

This Chapter describes the reference data editor (RDE) and its use with activated animation.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
General Information about the Reference Data Editor	596
Converting RDE templates	598
Changing signal states of a Located variable	600
Cyclical Setting of Variables	602
Unconditional locking of a section	605
Animation	606
Replacing variable names	608
Load reference data	609

General Information about the Reference Data Editor

At a Glance

Variables can be displayed in animation mode, 0x and 1x references can be blocked (forced) and unlocated element variables or elements of structures can be set cyclically using the Reference Data Editor (RDE). The behavior of the variables can be followed and modified online through directly accessing the variables and direct addresses used in the IEC program. Variable states are displayed in animation mode with different colors (disabled, cyclically set).

Maximum 250 entries are possible in the Reference Data Editor. If this limit is exceeded a warning message is generated when saving.

Creating RDE Templates

To create an RDE template, use the variables declared in the variable editor. There are various possibilities here:

If ...	Then
You make a double click on the corresponding numerical field in the first column,	you open the dialog **Lookup variables**, for selecting a declared variable or component of a structure.
You enter the variable names of a declared variable in the column **Variable name**,	the declared parameters are entered into the RDE template.
You enter the direct address in the column **Address**,	then the value, the format and in some cases the defined name of the corresponding signal are entered in the RDE template.
You use menu command **Insert Addresses...** to insert entire reference blocks into the column **Address**,	the values and the formats of the corresponding signals are entered into the RDE template.

Display Signal States

Stored signal states are always overwritten by the current values in the PLC with an activated animation (**Online →Animation**) when opening an RDE template.

The signal states in the PLC can be displayed in online mode using menu instruction **Controller status...** . When starting the PLC, you can view signal states corresponding with the program progress in animation mode.

Printing RDE Templates

To print an open RDE template, click in the **RDE** main menu on the menu instruction **Print**. An exact copy of the screen image of the RDE template will be created on paper.

NOTE: We recommend that you modify the printer properties to landscape paper format in the operating system (Windows). This will give you the complete image of the RDE template on a single page.

Using RDE Templates

Using an RDE template in more than one project is not recommended. This can cause doubled variable names to appear as well as variable names that did not exist in the original RDE template. The variables in the RDE templates are always displayed with the current reference addresses.

Converting RDE Templates

This procedure can be found in the description **Converting RDE Templates** *(see page 598)*.

Converting RDE templates

Introduction

RDE templates from an earlier version of Concept are automatically converted into the template format of the new Concept version. To differentiate between the converted RDE templates and the other RDE templates, they are saved with the file extension *.RDF.

⚠ **CAUTION**

Incomplete RDE templates are created!

Before the conversion make sure that the variables in the RDE template are declared in the opened project in the new version of Concept. New variables are listed in an error message and cannot be displayed in the RDE template (*.RDF) created from it.

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

Automatic Conversion

Automatic Conversion is performed when the RDE template of a previous version of Concept is opened:

Step	Action
1	Start the new version of Concept and open the project.
2	In the **Online** main menu click on the **Reference data editor** menu command. **Result:** The **RDE** main menu appears in the men bar.
3	In the **Online** main menu click on the **Reference data editor** menu command.
4	Select the directory, in which the RDE template (*.RDE) is saved (e.g. D:\CONCEPT_OLD). **Result:** All existing RDE templates (*.RDE or *.RDF) are displayed. **Note:** The files with the *.RDF extension come from the conversion of generated RDE templates (*.RDE).
5	Select the *RDE RDE template to be converted.
6	Click on the command button **OK**. **Result:** The **RDE AutoConvert** message appears. This informs you that the *RDE template was created in a previous version of Concept and is now being saved in a new format, so that it can be used in this version of Concept. The converted template is saved in a file with the *.RDF extension.
7	Click on the command button **OK**. **Result:** The converted RDE template (*.RDF) is displayed. **Warning:** All RDE template variables must be declared beforehand in the project. For new variables, the **RDE Template Errors** error message appears now, in which all faulty variables are listed. After closing the window, the converted RDE template opens, but only containing the declared variables.
8	Using the **Save reference data table under...** menu command, it is possible to save the converted RDE template in the directory in the new version of Concept (C:\CONCEPT_NEW). **Result:** The converted RDE template is stored in the Concept directory with the *.RDF file extension.

Changing signal states of a Located variable

Introduction

Located variables can be changed by checking the corresponding signal box in the column**Disable** and editing the value. Upon locking, the variable is separated from the hardware and is only used in the logic again if the disablement is undone. In this way, the changed signal states of all editors (FBD, SFC, LD, ST, LL984) are taken into account.

Unintended setting of values

Confirm values that were entered in an RDE table with the **Input** key. However, authorized values are also transferred if you switch to another input field using the cursor key or the mouse or if you leave the RDE table.

You can cancel an entry using the **ESC** key.

⚠ **WARNING**

UNINTENDED SETTING OF VALUES

Do not leave the RDE table (for example by clicking on another window) if you have already entered an authorized value in an input field, since otherwise the value will be transferred and unintended setting of values can occur.

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

Forcing inputs and outputs

When inputs are forced, signal states are transferred until the value in the RDE table is changed again. When outputs are forced, the new value appears at the beginning of each program cycle. When a subsequent change is made using the program logic, this value is not saved in the state RAM until the locking of the output has been removed.

⚠ **CAUTION**

All changed signal states are loaded directly onto the PLC.

Though not in the case of forced located variables.

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

Display of disabled variables

Variables that have been disabled by checking the check mark are shaded in color in the editor display. By removing the check symbol, the colored background of the corresponding variable is also no longer visible.

Loading reference data

Cyclically set values and disabled variables can be loaded onto the PLC using the menu command **Load reference data**.

These settings then remain the same until the user makes a change in the RDE Template, or the PLC loses the loaded data (e.g. by loading a different project).

NOTE: In an open RDE Template, the changed date is then automatically saved using the menu command **Load reference data**. The menu command **Save table** then no longer needs to be used.

Cyclical Setting of Variables

Introduction

Variables and structure elements can be changed by entering a set value corresponding to the data type of the variable in the**Set Value** column. This value will be written uniquely, if the corresponding signal's box in the **Cyclic Set** column is subsequently checked. The new signal state is loaded directly onto the PLC and is transferred to the cyclically set variables administrator. The signal state of the variable, attained after logic editing at the end of the cycle, is specified in the **Value** column. In animation mode, the cyclical setting of variables in IEC sections is displayed.

Unintended setting of values

Confirm values that were entered in an RDE table with the **Input** key. However, authorized values are also transferred if you switch to another input field using the cursor key or the mouse or if you leave the RDE table.

You can cancel an entry using the **ESC** key.

⚠ WARNING

UNINTENDED SETTING OF VALUES

Do not leave the RDE table (for example by clicking on another window) if you have already entered an authorized value in an input field, since otherwise the value will be transferred and unintended setting of values can occur.

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

Cyclic Set

NOTE: Cyclical setting of variables can only be performed ONLINE and in EQUAL mode, not in animation mode. Depending on logic, the displayed value may deviate from the cyclically set value.

When the cyclical setting check box is checked, the set value in the **Set Value** column can still be changed.

If the box in the column **Cyclic Set** is unchecked, the signal state in the column **Value** is loaded onto the PLC and is used in the logic.

A maximum of 300 variables can be cyclically set. For cyclical setting, the length of the entry is limited to 150 characters in the column **Variable Name**, because this name is sent to control. If a variable is used several times in the reference data editor, the most recently entered value will always be the one taken into account for cyclical setting.

NOTE: All changed signal states are loaded directly onto the PLC.

⚠ CAUTION

Modified variable names are not recognized by replacements.

When a variable is cyclically set, the spelling of the variable name should not be changed in the variables editor.

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

Cyclical setting and locking of signal states in the operating modes:

Mode	Option	Meaning
LOCAL	Disable	The variables declared in the Variable Editor can be written in the RDE Template in local mode. The signal states specified in online mode are displayed in local mode but cannot be changed and have no effect.
ONLINE	Disable	The changed signal states of located variables are transferred directly from the program logic.
LOCAL	Cyclic Set	Cyclical setting of variables cannot be executed in local mode.
ONLINE	Cyclic Set	The signal state in the column **Set Value** is used in logic editing by checking the box (check mark visible), and supplies a value at the end of the cycle, which is displayed in the column **Value**.

Getting/deleting cyclical set list

The cyclical values set in animation mode can be inserted into the RDE Template in disabled animation using the menu command **Get CSL**.

Cyclically set values are recognized in the RDE Template by the check mark in the column **Cyclic Set**, and are automatically recognized row by row. It is therefore referred to as a cyclical set list. Using the menu command **Online →Get CSL** this recognized list will be inserted dependently from the selected row in the RDE table. Getting and inserting the cyclical set list can be done as often as required. The most recent cyclical set list is always located on the clipboard and can only be deleted using the menu command **Delete CSL**. Thereafter, getting and inserting is no longer possible until values are cyclically set at the next animation.

NOTE: Each time, the system gets**all** cyclically set values .

Loading reference data

Cyclically set values and disabled variables can be loaded onto the PLC using the menu command **Load reference data**.

These settings then remain the same until the user makes a change in the RDE Template, or the PLC loses the loaded data (e.g. by loading a different project).

NOTE: In an open RDE Template, the changed date is then automatically saved using the menu command **Load reference data**. The menu command **Save table** then no longer needs to be used.

Unconditional locking of a section

At a Glance

At the section to be inhibited, the logic must carry a BOOL data type "output" and it should be noted that the section is disabled at configured "1".

⚠ **CAUTION**

Risk of unwanted process states.

Locking a section does not mean that programmed outputs within the section are deactivated. If an output was already set during a previous cycle, this state also remains after the section has been inhibited. It only ceases to be possible to change the state of these outputs once the section has been inhibited.

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

NOTE: A section that contains a logic to lock/release other sections should not be disabled, if possible. Output state disabled sections cannot be changed.

Procedure for unconditional locking of a section.

The following procedure is performed to disable a section unconditionally in the RDE table:

Step	Action
1	By double-clicking in a text box in the first column in the table (1 … 100) open the dialog box **Look up variables**.
2	In the zone **Data type** select the option button **Structured** and from the list select **SECT_CTRL**. **Reaction:** The names of all sections are displayed.
3	Select the name of the file to be disabled and using the command button **Elements...** open the dialog box **Select elements by type**.
4	Select the line **disable : BOOL** and confirm with **OK**. **Reaction:**The structured variable (Sectionname.disable) to which the section to be disabled is assigned, is entered in the RDE table.
5	Link the PLC and the programming device (**Online** →**Link...**), and load the user program onto the PLC (**Online** →**Load...**). **Reaction:** The PLC is in ONLINE and ANIMATIONS mode.
6	In the column **Value** enter a configured "1". **Reaction:** The section is disabled and will not be processed.

Animation

At a Glance

Animation can only take place in ONLINE mode. By activating Animation in the Reference data editor it is possible to display the signal states of the variables, and to observe the behavior of the output signals while the program is running.

During animation, signal states can be changed online also. The new values are automatically loaded onto the PLC and are taken into account during the next cycle.

NOTE: When changing a value it should be ensured that the locking of the variable is subsequently removed. It is impossible to animate disabled variables correctly.

Animation status

The column **Animation status** specifies the status of entered unlocated Variables during animation.

This table provides an overview of the animation status possibilities:

Display	Mode	Cause
Not used **Note:** In LOCAL mode, this display changes to "Unequal program"	ONLINE, ANIMATED	A variable not used in the user program, which is declared in the Variable Editor, was entered in the RDE table.
Inhibited I/O flag bits	ONLINE	An unlocated variable was cyclically set during the ANIMATIONS mode.
Unequal program	ONLINE	A variable that is used in the user program, which is declared in the Variable Editor, was entered in the RDE table. The program is in MODIFIED mode.
Unequal program Note: In ONLINE mode, this display changes to "Not used".	LOCAL	A variable not used in the user program, which is declared in the Variable Editor, was entered in the RDE table.

Display of forced and cyclically set signals in ANIMATIONS mode

The variables that are forced or cyclically set in the reference editor are labelled with a colored background in the individual editors.

Forced variables are displayed in the following way:

Editor	Display
IEC editors (FBD, LD, SFC, IL, ST)	When forcing occurs, variable names are shaded in ochre (brown-yellow).
LL984 editor	When forcing contacts, variable names are underlined. When forcing spools, an opened contact ("inhibited") is displayed before the spool.
Monitoring fields and **Display** dialog	When forcing occurs, variable names are shaded in ochre (brown-yellow).

Cyclically set variables are displayed in the following way:

Editor	Display
IEC editors (FBD, LD, SFC, IL, ST)	When cyclical setting occurs, the variable name is shaded in violet.
Monitoring fields and **Display** dialog	When cyclical setting occurs, the variable name is shaded in magenta.

NOTE: In LD (Ladder Diagram) spools and contacts are displayed in color. However, due to forcing and cyclical setting, it is possible that the colors of the variable names will be different from the color display of spools and contacts.

Display of forced and cyclically set element structured variables in ANIMATIONS mode

If a structured variable element is forced or cyclically set, there are different display possibilities.

Display	Cause
The name of the structured variable (e.g. motor) is shaded in color.	In the editor, a multi-element variable (e.g. motor) is displayed, in which one or more elements is forced or cyclically set.
The name of the structured variable element (e.g. right motor on) is shaded in color.	In the editor, a forced or cyclically set element of a multi-element variable (e.g. right motor on) is displayed.
The name of the structured variable element (e.g. right motor on) is shaded in color, but the name of the element is not.	In the editor, an element of a multi-element variable that is not forced or cyclically set is displayed, but a different element of this multi-element variable is cyclically set or forced.

Replacing variable names

At a Glance

When using an open RDE table it is possible to simultaneously edit the Variable Editor. If variable names are changed in the Variable Editor using the Find/replace function, these changes are automatically adopted in the open RDE table. In this case the RDE animation is initially terminated and the RDE table must be reloaded.

Procedure and reaction

For the automatic adoption of replaced variable names in the simultaneously open RDE table, the following steps are to be performed:

Step	Action
1	Open a section and create an online link. **Note:** The state between PLC and programming device must be EQUAL. If not, load the program into the PLC.
2	Start the animation (**Online** →**Animate binary values**). **Reaction:** The signal states of the section are displayed in color.
3	Open an existing RDE table (**RDE** →**Open table...**). **Reaction:** The RDE animation is started.
4	Open the Variable Editor (**Project** →**Variable declaration...**).
5	Using the command button **Find/replace** open the dialog **Find/replace**.
6	Replace an existing variable name with a new name (Command button **Replace**). **Reaction:** The variable name was changed in the Variable Editor.
7	Exit the Variable Editor using **OK**. **Reaction:** The section is automatically updated, and the RDE animation is terminated.
8	Close the RDE table and save the changes (Command button **Yes**).
9	Reopen the saved RDE table (**RDE** →**Open table...**). **Reaction:** The RDE animation with the changed variable name is recovered.

Load reference data

At a Glance

In the same cycle, the variables changed in the reference data editor are sent to the PLC, using the menu command **Online** →**Load reference data**.

NOTE: To perform the loading, the animation must be disabled.

ASCII Message Editor

19

Introduction

This chapter describes the ASCII message editor.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
19.1	ASCII Editor Dialog	612
19.2	User Interface of ASCII Message Editor	621
19.3	How to Continue after Getting a Warning	628
19.4	ASCII Editor in Offline/Combination/Direct Modes	629

19.1 ASCII Editor Dialog

Introduction

This section describes the ASCII editor dialog.

What's in this Section?

This section contains the following topics:

Topic	Page
Generals to ASCII editor dialog	613
Text	614
Variables	615
Control code	616
Spaces	617
Carriage Return	618
Flush (buffer)	619
Repeat	620

Generals to ASCII editor dialog

Introduction

Use the ASCII message editor to create, edit, and simulate ASCII messages. The ASCII message text/control that is created in the editor can be transferred to the selected PLC. Conversely, the ASCII messages internal to the controller can be uploaded to the editor.

An ASCII message set consists only of a list of messages that satisfy certain rules. The number of messages allowed and the maximum length of the ASCII message set is defined as part of the PLC configuration. Each message consists of a list of ASCII message fields separated by commas.

The following fields are currently supported:

- *Text, page 614*
- *Variables, page 615*
- *Control code, page 616*
- *Spaces, page 617*
- *Carriage Return, page 618*
- *Flush (buffer), page 619*
- *Repeat, page 620*

Preconditions

This function is only available when using:

- Concept for Quantum
- The modules J892 or P892
- Programming language LL984

Text

Introduction

The text messages defined by text fields take the format `'Hello World'` whereby `Hello World` becomes the text to be forwarded. The single quotation marks are the delimiters. The ASCII message editor development dialog provides a development area and a simulator area where the composed message is interpreted and displayed for you to make any necessary edits before leaving the editor dialog.

Message Length

An ASCII message can be as long as 134 words. Three words are for overhead plus the actual message maximum of 131 words (2 characters per word).

Message words are used up as follows:

Field type	**Field length (in words)**
ASCII text	1 + length of text / 2 rounded up
Return	1
Flush 0, 1	1
Flush 2, 3	2
Control	1
Variable	1
Repeat	2
Space	1

Variables

Introduction

A variable will be given the format NTF.

The meaning of this is:

- N representing the decimal number (1...99) of the data fields of the data type defined by T.
- T is the data type of the variable.
- F the decimal field width for the variable.

Data Types

The data types supported are:

Type	Repetition factor
A = ASCII character	1
B = binary number	1 to 16
H = hexadecimal	1 to 4
I = integer	1 to 8
L = integer with leading 0s	1 to 8
O = octal	1 to 6

Example

For example: 2H2 means:

- 2 registers (N)
- in hexadecimal (T)
- containing 2 hexadecimal numbers (F)

N can fit into the number of data registers needed, but it is not an absolute requirement.

The relationship is:

Type	Relationship
A	Number of registers = N/2 (next upper integer value)
B	Number of registers = N

H	for 1 ≤F ≤4... Number of registers = N
	for 5 ≤F ≤8... Number of registers = 2 x N

I and L	The same as H
O	Number of registers = N

Control code

Meaning of Control Code

A control code is given the format "Null", with Null being a three characters OOO, and the double quotation marks are delimiters.

For example: "017"

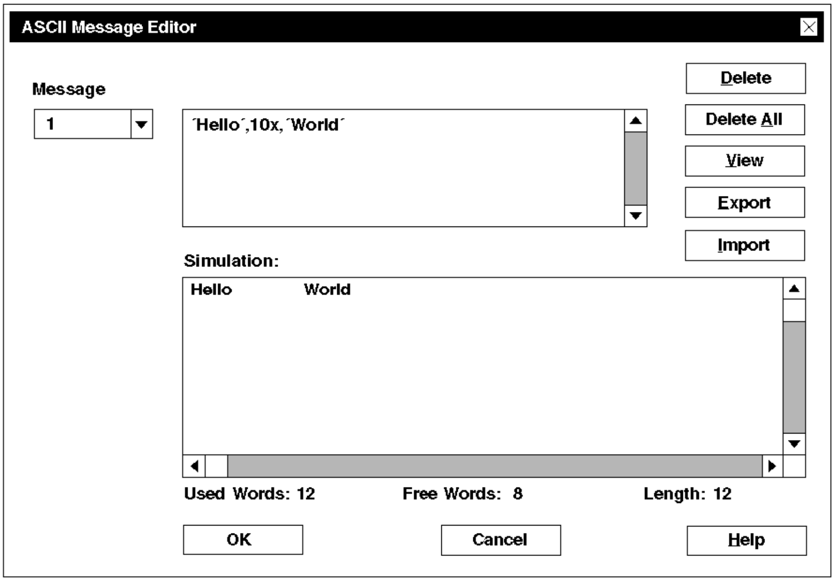
Spaces

Meaning of Spaces

A space field is given the format ddx, with dd being a decimal number (1..99) used to determine how many spaces are to be added to the message.

Representation of Dialog

Many spaces between text:



´Hello´,10x,´World´

Simulation:

Hello   World

Used Words: 12

Free Words: 8

Length: 12

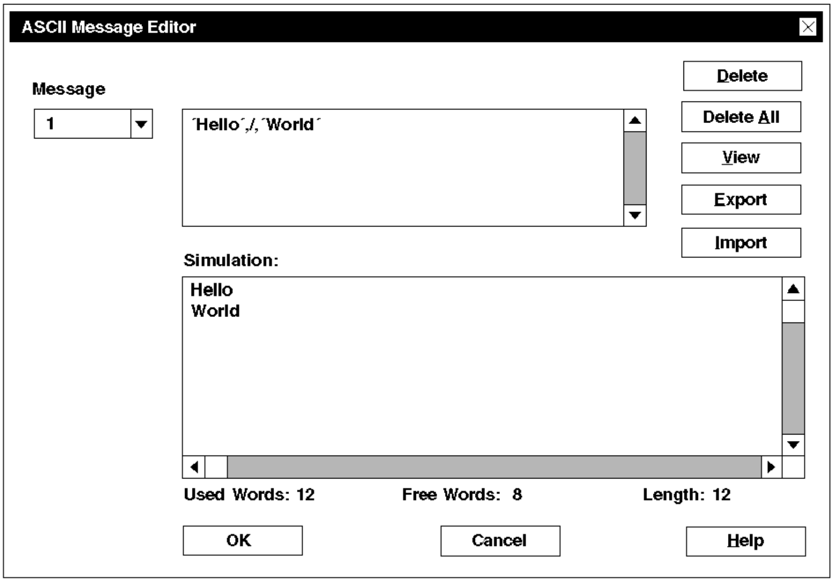
Carriage Return

Meaning of Carriage Return

A carriage return field will add a carriage return to the output information, and it has the format, /.

Representation of Dialog

Carriage return:



Message: 1

´Hello´,/,´World´

Simulation:

Hello
World

Used Words: 12   Free Words: 8   Length: 12

Flush (buffer)

Meaning of Flush

This will expressly specify for the P892 only how the input message buffer is to be cleared. This field has the format <*>/.

The * can be any of the following:

*	**Meaning**
0	Remove all characters in the buffer. An example is: <0> clears all.
1;bbb	Removing the number of characters specified by bbb, whereby bbb is a number (1...255). For example, <1;100> flushes the first 100 characters in a buffer.
2;hhhh	Scanning the message for the 2 characters that are specified by the hexadecimal numbers hhhh. If a match is found, delete all characters up to but not including the match. An example is: <2;5445> causes the buffer ‘12TEST’ to become "TEST".
3;rrr;hhhh	Scanning the message for the 2 characters that are specified by the hexadecimal numbers hhhh. If a match is found, delete all characters up to and including the match. The search is to be performed as often as specified by rrr, whereby rrr is representing a decimal number 1...255. **Example:** <3;2;5445> causes the buffer ‘12TEST3456TEST789TEST’ to become ST789TEST.

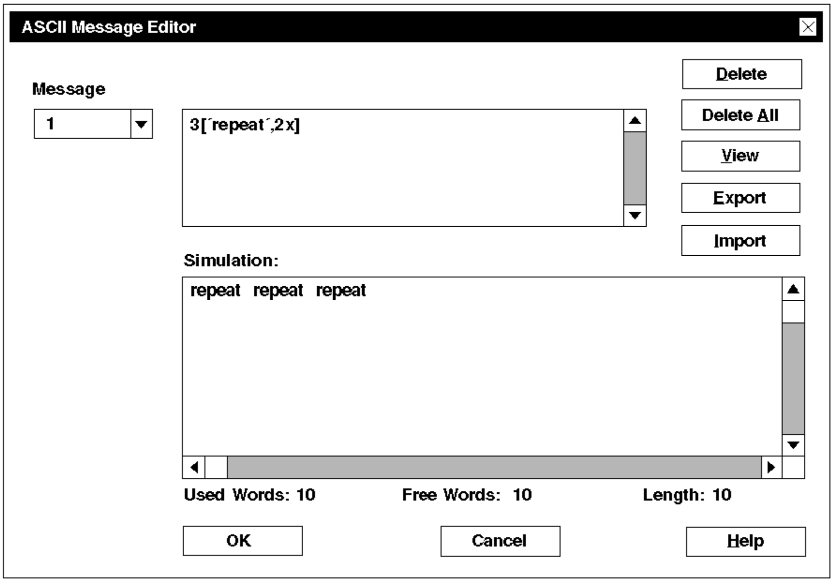
Repeat

Meaning of Repeat

Use this message field to specify that a number of message fields will be repeated several times. This field has the format dd(*), with dd being a decimal repetition factor (1....99), ( ) are delimiters, and * is a series of message fields.

Representation of Dialog

Repeated text:



ASCII Message Editor

Message: 1

3[´repeat´,2x]

Simulation:

repeat repeat repeat

Used Words: 10   Free Words: 10   Length: 10

19.2 User Interface of ASCII Message Editor

Introduction

This section describes the user interface of the ASCII message editor.

What's in this Section?

This section contains the following topics:

Topic	Page
How to Use the ASCII Message Editor	622
Message Number	624
Message Text	626
Simulation Text	627

How to Use the ASCII Message Editor

Invocation of ASCII Message Editor

The ASCII message editor is invoked from the **ASCII messages...** menu item in the **Project** menu. This editor allows you to add/modify/delete messages in a temporary work space, then save or cancel the changes.

Add New Messages

To add a new message, type the new message number into the **Message** text box and type a syntactically correct message into the message text box. As you enter a message into the message text box, its corresponding simulation is displayed in the **Simulation** text box. When the message is syntactically incorrect, it is displayed in red.

Modify Existing Messages

To modify an existing message, select a message from the **Message number** list and change the text.

Delete Messages

To delete a message, select a message from the Message number list and click on **Delete**.

Clicking on the button **Delete All** will remove all messages in the temporary workspace. The button is active if there is at least one ASCII message in the message set. Selecting this option results in the display of a confirmation dialog.

View

Clicking at the button **View** will produce a view of the displayed **ASCII message** dialog. The view message format is message number followed by message text.

You can select from the choices available. To download the editor from the view list, click on the message and on **OK**.

Save Changes

Use the button **OK** to save processes performed while working with the ASCII editor and to close the dialog. Each message that has been created or changed is checked for syntactic correctness at this point. The checking begins at the current message and wraps around until all messages are checked. If a syntax error is detected, a definition of the error is displayed first, and as soon as the error dialog is cleared, the message itself appears with the cursor on the faulty character. Every attempt to add ASCII characters which will cause the size of the entire message area set in the configuration to be exceeded will generate an error.

Length, Used and Free

These fields display the length of the current message (in words), the number of words used and the number of words remaining.

Message Number

Introduction

The combo box **Message number** is a dialog that contains a message selection list with a check mark next to the currently selected message.

Use this dialog to select existing message numbers and/or to add new message numbers. As long as there are no messages, text box and list are empty. If there are messages, the editor is initially displayed with the text box containing the first message number and a list of message numbers for existing messages. The message number that relates to the currently displayed message is posted above the list box.

Action

For the selection of an existing message, click at the list button and mark a number in the list or type the number into the text field. A new message number can be inserted by typing the number into the text field.

Effects

If the message number assigned to an existing message is changed (either text or list entry), the text box **Message** will display the message text for the message number and the box **Simulation** shows the simulation of the message. If a new message number has been entered, the text boxes **Message** and **Simulation** will be cleared.

Error handling

The following errors can be appearing:

If...	**Then ...**
an unauthoried character is entered in the number field of the message.	a message field dialog will show: "Message number contains illegal characters". After acknowledging the error, the message number is reset and the process will continue in the text box **Message**.
the text box **Message** is not filled out.	a message field dialog will show: "`There must be a message number before text can be entered`". After acknowledging the error, the message number is reset and the process will continue in the text box **Message**.
the number is greater than the maximum number set in **Configure →ASCII Setup...**.	a message field dialog will show: "` Message number exceeds maximum set in configuration`". After acknowledging the error, the message number is reset and the process will continue in the text box **Message**.

Message Text

Introduction

The text box **Message** is a text editor with free format for the entry of ASCII messages. This editor allows one arbitrarily long line of free-format text. Although the text should follow the ASCII message syntax, it does not necessarily have to be syntactically correct prior to activating the **OK** button, even though a view note regarding validity will appear already during entry of the messages.

Actions

A currently selected message is made available for editing, otherwise a new message can be entered. The standard Windows edit operations (**Cut**, **Paste**, **Copy**, ...) are allowed.

Effects

If the message is syntactically correct, its text will be displayed in normal textual color, if not, the display will be in red. Text wraps so there is never a case where horizontal scrolling is required.

Simulation Text

Introduction

The text box **Simulation** is a read-only multi-line field. The simulated output of the current message is displayed in this window. As messages are added or changed, the simulated output is displayed in the simulation window.

Special Considerations

The simulation of control codes is shown as the ASCII character that corresponds to the controller, except those control codes that are not authorized in Windows text control and are written as an ’l’.

NOTE: Any simulation greater than 32 k characters is truncated to this maximum.

19.3 How to Continue after Getting a Warning

How to Continue after Getting a Warning

Introduction

A few conditions will allow continuing work with the ASCII editor although with possibly restricted functionality.

NOTE: To match a configuration, messages may be deleted.

Exceeding the Total Messages

Message numbers that are above the maximum limit set in **Configure →ASCII Setup...** will only be available for display or delete. These messages appear grayed out.

The accompanying warning reads: "`Warning: Some message numbers exceed the highest message number xx, defined in Configure. All messages beyond xx can only be displayed or deleted.`"

Exceeding the Message Area Size

If the size of the message in the data base is greater than the size defined in **Configure →ASCII Setup...**, a warning will appear. You can continue to view, change, or delete but changes cannot be saved unless the size falls below the configuration setting.

This warning reads: "`Warning: The size of the ASCII message area, xx, exceeds the maximum size, xx, defined in Configure.`"

Tips

NOTE: To match a configuration, messages may be deleted.

NOTE: Information about the ASCII character set can be found in the PLC User's Guide.

19.4 ASCII Editor in Offline/Combination/Direct Modes

ASCII Message Editor in Offline/Combination/Direct Modes

Offline

When using Concept to program in offline mode, the ASCII message editor is displayed with the set of messages saved in the data base. By activating the **OK** button, these messages will be saved in the database.

Direct

When using Concept to program in direct mode, the ASCII message editor will be displayed with the set of messages saved in the controller. By clicking on the **OK** button, the changes made to the ASCII messages will be downloaded to the controller.

Combination Mode

When entering the Combination mode, Concept checks whether the information in the controller matches the information in the data base. If a match is found, the controller is considered **EQUAL** to the database. A mismatch is marked as **NOT EQUAL**. If the status is **EQUAL**, the ASCII message editor will be displayed with the ASCII message set taken from the data base. If a displayed editor message is changed, these changes will be saved to the database and the controller after clicking the **OK** button.

Online functions

20

Overview

This chapter describes the various online functions.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
20.1	General information about online functions	632
20.2	Connect to PLC	633
20.3	Setting up and controlling the PLC	649
20.4	Selecting Process information (status and memory)	663
20.5	Loading a project	670
20.6	Section animation	681
20.7	Online Diagnosis	685
20.8	Logging Write Access to the PLC	687

20.1 General information about online functions

General information

At a Glance

After setting up a link via Modbus, Modbus Plus or TCP/IP between the programming device and the PLC the project can be loaded onto the PLC. Now special online functions for displaying and changing the current value in the PLC state RAM are available in the separate editors. The PLC can be controlled.

⚠ CAUTION

A communication timeout or a general memory protection failure could occur if the system clock of the programming device is changed while it is online.

If the running program cannot be terminated, all animated program sections should be closed , or the animation should be turned off in order to reduce the possibility of getting into a time critical operation.

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

20.2 Connect to PLC

Overview

This section contains information about connecting the PLC.

What's in this Section?

This section contains the following topics:

Topic	Page
General	634
Presettings for ONLINE operation	637
Modbus Network Link	638
Modbus Plus Network	639
Modbus Plus Bridge	644
TCP/IP-Network Link	646
Connecting IEC Simulator (32 bit)	647
State of the PLC	648

General

Introduction

A connection can be created between a programming device and the PLC.

In monitor operation (it is not supported by M1E PLCs), it is possible to make alterations to the IEC sections, but these cannot be downloaded to the PLC. When exiting Concept a warning will be displayed.

NOTE: Only one programming device may be connected to the PLC.

Limited PLC Login

When logging into the PLC, the following restrictions are imposed for Quantum CPUs 140 434 12 A and 534 14 A/B:

- If a programming device is already connected with the PLC in programming mode, then no other programming devices can be connected with the PLC.
- If a programming device is already connected with the PLC in Monitor mode then other programming devices can be only connected with the PLC in Monitor mode. An attempt to connect with the PLC in another operating mode is not possible for the other programming devices.

Consistency check

If a project is open and a connection between the programming device and the PLC is to be created, a consistency check is automatically carried out between the program, EFBs and DFBs on the programming device, and the PLC. The result of this check (**EQUAL**, **MODIFIED** or **NOT EQUAL**) is displayed in the status bar and written in a file. This file is located in the project directory of Concept and has the name PROJEKTNAME.RMK. It functions for internal usage and automatically changes its contents. The meaning of the individual entries can be found in the following diagram.

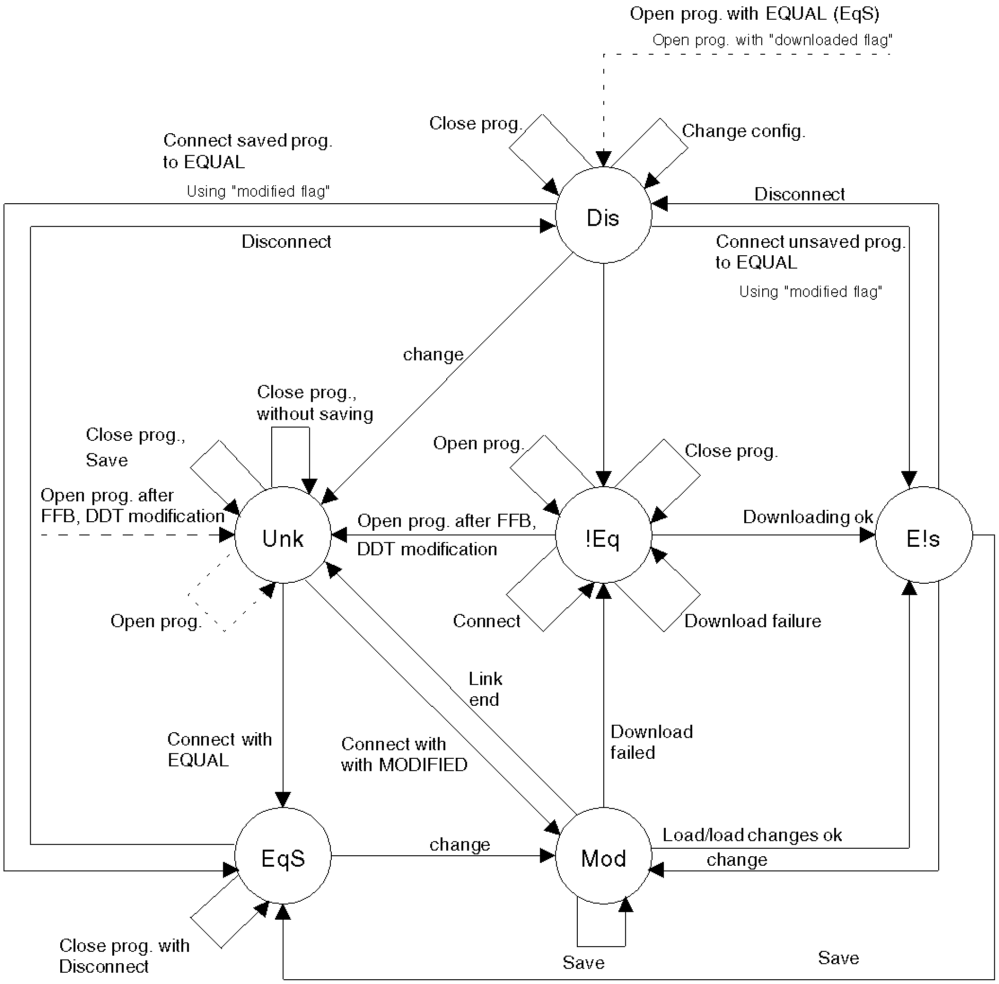
Status descriptions

Status descriptions:

- **EQUAL**
  The program on the programming device and the PLC is consistent.
- **NOT EQUAL**
  The program on the programming device and the PLC is not consistent. To establish consistency use the menu command **Online** →**Download...** .
- **MODIFIED**
  The program on the programming device was modified. The modifications can be made online in the PLC with the menu command **Online** →**Download changes**.
  Note: Even when changes that are not relevant to the code (e.g. creating/changing comments in IL/ST, moving objects (without affecting logic) exist in FDB/LD/SFC), the **MODIFIED** status is displayed temporarily. When the section is next analyzed (**Project** →**Analyze project**, **Project** →**Analyze section** or **Online** →**Download changes**), the program automatically reverts to the **EQUAL** status (if no changes have been carried out that are relevant to the code). Even if changes that are relevant to the code have been carried out, only these sections appear in the **Download changes** dialog box.

Relationships between states

The diagram shows the relationships between the different program states:



Unk UNKNOWN
Dis DISCONNECTED
!Eq NOT EQUAL
Mod MODIFIED
E!S EQUAL but not saved
EqS EQUAL and saved

Presettings for ONLINE operation

At a Glance

The dialog box **Link PLC to** can be used to specify settings for both the PLC link and ONLINE mode resulting from it.

Access

It is possible to specify which functions will be executed in the ONLINE operation, i.e. which menu commands are available in the **Online** main menu.

Protocol types

To link the programming device and PLC, it is important to know which network the communicating nodes are in so that the correct protocol type is selected.

Use the table to decide which protocol type fits the network link used:

Linking the network nodes	Protocol type
Serial Interface	Modbus
SA85-/PCI85-Adapter	Modbus Plus
NOE-module (on Ethernet-Bus SINEC H1)	TCP/IP
TCP/IP Interface map (32-Bit Simulation)	IEC Simulator (32-Bit)

NOTE: The programming device can always only be linked to one PLC. This means that before a link is made to another PLC, any existing link must be terminated with the **Terminate Link** menu command.

Modbus Network Link

Introduction

For a Modbus network link, the settings of the modbus interface must correspond with the settings on the PLC.

The interface is edited in the **Modbus Port Settings** dialog (**PLC Configuration** → **Modbus Port Settings...**).

Protocol Settings for Modbus

When the Modbus protocol type is selected, specify further data in the **Protocol Settings: Modbus** range. Specify the Node Address (Node No.) on the PLC and enter this in the corresponding text box. You can determine the transfer mode for communication between the PLC and the host computer.

The following modes are available according to the application:

Application	Mode
Communication with various host devices. The ASCII mode works with 7 data bits.	ASCII
Communication with an IBM compatible personal computer. The RTU mode works with 8 data bits.	RTU

After the serial interface for the Modbus network link has been specified, using the **Settings...** command button, open the **Settings for COMx** dialog. Enter the settings for the interface here, as in the **Modbus port settings** dialog.

Use the **OK** command button to create the ONLINE link.

Modbus Plus Network

Introduction

For a Modbus Plus network connection, enter in the **Protocol settings: Modbus Plus** range whether the 16-Bit IEC-Simulator (**Port 0**) or the Modbus Plus interface (**Port 1**) is being used.

All nodes on the local network are displayed in the list. Additionally, the routing path of the token rotation sequence in the network, which can contain up to 5 Node addresses is displayed. Up to 64 nodes can be addressed on one network, i.e. a routing path address can be between 1 and 64. Several networks can be linked via a bridge.

NOTE: The node list of a different network can be displayed by double-clicking on a listed bridge.

To pass the program execution to the Modbus Plus device driver, Concept triggers a MS DOS Software Interrupt. The preset Interrupt number for this is 5C (hex).

NOTE: If no virtual Modbus Plus driver is installed, the virtual MS DOS environment in Windows NT has problems when reacting to the software interrupt. If a share violation (Exception) occurs under certain conditions, change the Interrupt number to 5D (hex) in the MODICON.INI file:

```
[PORTS]

mbp0=5d
```

If the Interrupt 5D from the NTVDM.EXE is activated the share violation should no longer occur.

IEC Simulator (16 Bit)

The simulator simulates a coupled PLC via Modbus Plus. The address of the programming device is displayed in the list in the routing path.

The simulator is active if in the **Protocol settings: Modbus Plus:** area, the option **Port 0** is selected.

NOTE: When the simulator is active, no further nodes can be displayed.

The simulator is only available for the IEC languages (FBD, SFC, LD, IL and ST).

PLC as Modbus Plus Node

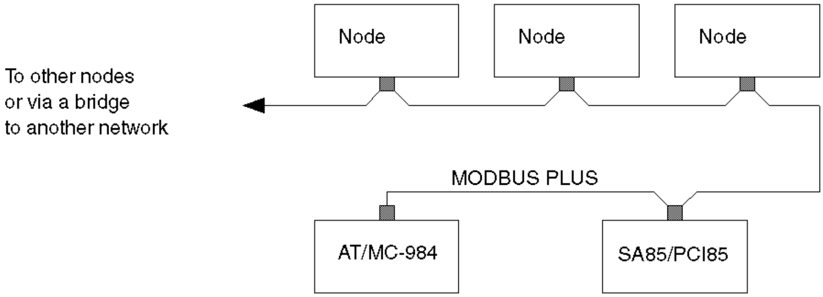
When the PLC is a Modbus Plus node, the address which the PLC has in the routing path is displayed in the list. This address corresponds to the node address which is set with a rotary switch on the back of the CPU.

SA85/PCI85 as Modbus Plus Node

The SA85 module is a Modbus Plus adapter for an IBM-AT or compatible computer.

The port address is displayed in the list. The address shows in which network the SA85/PCI85 is installed.

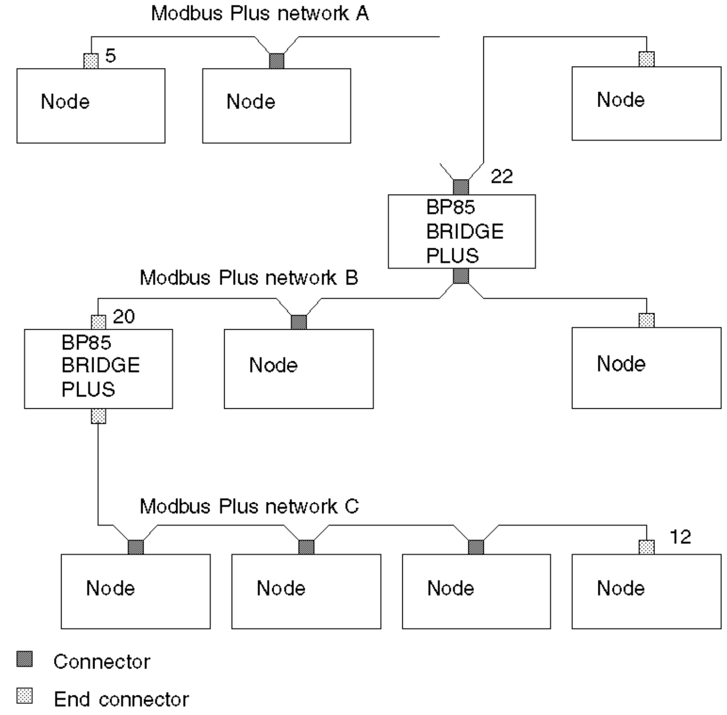
Displaying a routing path with SA85/PCI85:



Bridge Plus as Modbus Plus Node

A Bridge Plus (BP85) links nodes within two Modbus Plus networks. The Bridge is displayed in the list box, and the next Modbus Plus network can be accessed by double-clicking on the Bridge.

Displaying a routing path with a Bridge Plus BP85:



Example:

The example shows a routing path across 3 Modbus Plus networks. The task is to send a message from node 5 in network A to node 12 in network C.

The routing path here is 22.20.12.00.00 and it is made up as follows:

Path	Meaning
22	The first address contains the network A Bridge Plus address from Network A from output node 5. This means the message is sent from output node 5 across this Bridge to the next network, B.
20	The second address contains the Bridge Plus address of the next network, B. Here, the message is sent from network B to the third network, C.
12	The third address contains the address of node 12, the destination segment.
00.00	Addresses four and five are set to 0 because there are no further forwarding addresses.

Bridge as Modbus Plus Node

A link between the Ethernet and the Modbus Plus network or between two Modbus Plus networks is created via the Modbus Plus Bridge.

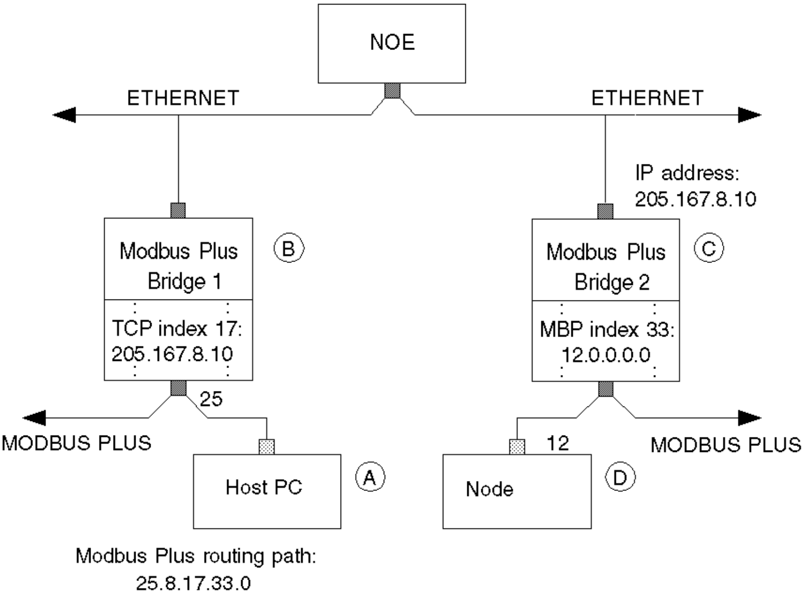
The Modbus Plus Bridge should be regarded as a host computer and must be configured in the **Protocol settings: TCP/IP** area. Enter the IP address or the host name of the Bridge, and finally in the text box **Protocol type:** change to Modbus Plus network setting.

The Modbus Plus Bridge is only listed in the list of nodes in the Modbus Plus network as a host name which was previously entered in the **Protocol Settings: TCP/IP** area. A double-click on the corresponding host name opens the **Modbus Plus Bridge** dialog box for 5 byte routing path configuration.

Further action in the dialog box can be found in the chapter"*Modbus Plus Bridge, page 644*".

Example:

In the dialog box **Modbus Plus Bridge**, create the routing path 25.8.17.33.0, which defines the following link (from A to D):



A The message sent from the host computer contains the 5 byte Modbus Plus Routing Path. The first byte with the node address of the host computer refers to the Modbus Plus Bridge linked to it. The Modbus Plus Bridge 1 receives the message on internal path 8, as specified in byte 2.

B The TCP Index No. 17 (byte 3) administered in the Modbus Plus Bridge passes the message on to the configured node with the IP address 205.167.8.10. In this case the node with this IP address is another Modbus Plus Bridge.

C Modbus Plus Bridge 2 receives the message. The MBP Index No. 3 given in 4 byte and administered by the bridge passes the message on to the configured Modbus Plus node. In this case the node 12.0.0.0.0.

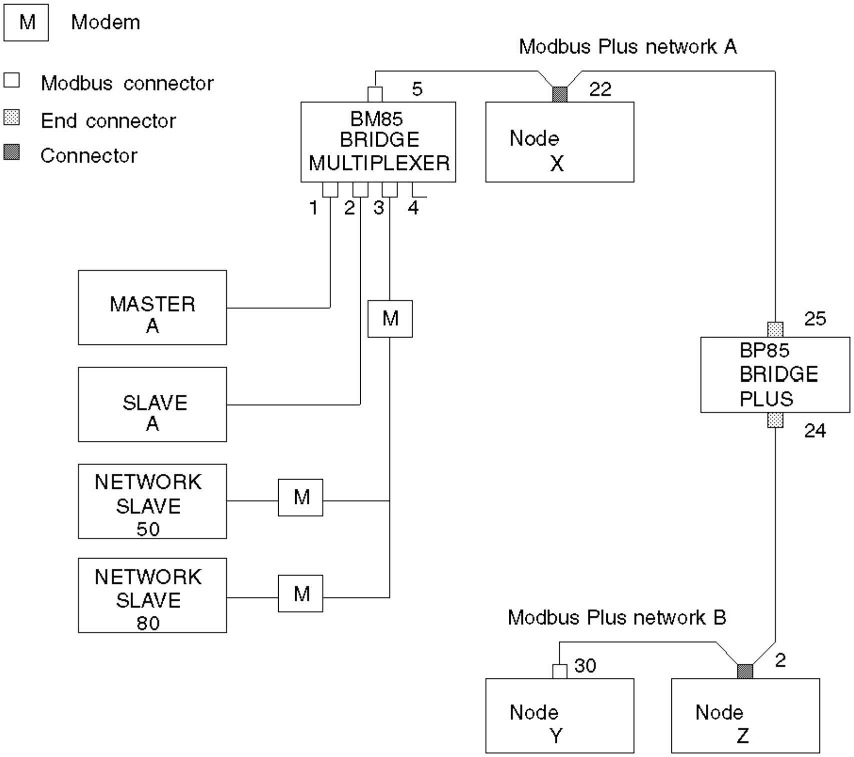
D The message has reached its destination, Modbus Plus node 12.

Bridge Multiplexer as Modbus Plus Node

The BM85 Bridge Multiplexer links up to four Modbus devices or Modbus networks to a Modbus Plus network.

See also "User's Guide BM85 Modbus Plus Bridge/Multiplexer."

Displaying a routing path with a Bridge Multiplexer BM85:



Modbus Plus Bridge

At a Glance

Enter the 5 byte routing path which defines the link from the host computer to the Ethernet node in this dialog box.

Making settings

The following table describes how to define the routing path:

Setting zone	Routing path byte	Meaning
Bridge Path	2. Byte	A maximum of 8 links can go out from the Bridge to the other network, and one of these should be selected.
IP routing byte	3. Byte	Enter an index no. which is assigned to an IP address. This IP address should correspond to an Ethernet node address where the message is then sent. If this IP address is being sent to another Modbus Plus Bridge in the Ethernet,another node address (MB+ routing byte) must be given for it to be transferred further into the Modbus Plus network.
MB+ routing byte	4. Byte	If a link is displayed between two Modbus Plus networks via two Modbus Plus bridges, the index no. of the Modbus Plus node must be entered here. This index no. is also assigned to a node address. If there is no link across a different bridge, the value "0" is entered.
Complete address	5. Byte	The whole 5 byte routing path is displayed according to the setting. The first byte is then automatically adjusted to the node address of the host computer.

Modbus Plus index no.

The assignments of the Modbus Plus index no. are pre-set and can be selected between 0 and 255. Note that index no. 255 is reserved for specific operations. When this index no. is selected, data selection or loading is permitted between a TCP/IP node and the Modbus Plus Bridge via an internal command. Index nos. 250 – 253 are reserved and cannot be used.

The index in the Modbus Plus routing path is shown in the following table:

Index	Modbus Plus routing path
1 ... 64	1.0.0.0.0 ... 64.0.0.0.0
65 ... 128	2.1.0.0.0 ... 2.64.0.0.0
129 ... 192	3.1.0.0.0 ... 3.64.0.0.0
193 ... 249	3.2.1.0.0 ... 3.2.57.0.0

TCP/IP Index No.

The assignments of the TCP index no. follow automatically after the IP address of the Modbus Plus Bridge has been specified in the **Link** →**Protocol Settings: TCP/IP** dialog box. Each index is assigned to an IP address where the first 3 bytes are assigned to the first 3 bytes of the Modbus Plus Bridge IP address. The 4th byte is counted upwards from 1 to 255 at the most.

Example:

For a Modbus Plus Bridge IP address of 205.167.4.65, the TCP/IP addresses are automatically pre-set, as in the following table:

Index	IP address
1	205.167.4.1
2	205.167.4.2
...	...
255	205.167.4.255

NOTE: Refer to the "174 CEV 200 30 TSX Momentum Modbus Plus to Ethernet Bridge User Guide" for a detailed description of the Ethernet Bridge.

TCP/IP-Network Link

Introduction

For an Ethernet link, select the protocol type TCP/IP in the **Connect to PLC** dialog box.

Protocol Settings for TCP/IP

To connect to other Ethernet nodes, specify the IP address or the host name of the Ethernet node in the **Protocol Settings: TCP/IP** range.

To connect to the Ethernet via Modbus Plus node, specify the IP address or the host name of the Modbus Plus Bridge in the **Protocol Settings: TCP/IP** range (see also "Bridge as Modbus Plus Node *(see page 642)*").

Connecting Quantum to the Ethernet

You can connect the Quantum to the Ethernet Bus by configuring the NOE module. By doing this, it is possible to communicate with other automation components in the Ethernet Bus system via the host computer.

Connecting IEC Simulator (32 bit)

Introduction

The simulator simulates a PLC connected via TCP/IP, where the signal status of the I/O modules can also be simulated. Up to 5 host computers are connected to the simulated PLC at the same time.

To activate the simulator, select the protocol type IEC simulator (32 bit) in the **Connect to PLC** dialog box.

Protocol Settings for IEC Simulator (32 bit)

The simulator is active, if you specify the address of your TCP/IP interface board in the **Protocol Settings: IEC Simulator (32 bit)** range.

The TCP/IP address can be obtained on the title bar of the Concept simulator program PLCSIM32.

NOTE: At the present time the simulator is only available for IEC languages (FBD, SFC, LD, IL and ST).

State of the PLC

At a Glance

With a network link, the state of the PLC is displayed in the list of nodes in the Modbus Plus network in the **Link to PLC** dialog box.

States of the PLC

All the states which can arise are listed in the following table:

State	Meaning
Running	Identifies a PLC with a program running.
Stopped	Identifies a PLC with a program which has stopped.
Unknown	Identifies an unknown PLC.
Not configured	Identifies a PLC without a hardware configuration, i.e. no online functions are possible.

20.3 Setting up and controlling the PLC

Overview

This chapter contains information about setting up and controlling the PLC.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information	650
Setting the Time for Constant Scans	651
Single Sweeps	652
Deleting memory zones from the PLC	654
Speed optimized LL984-Processing	655
Save To Flash	656
Reactivate flash save	659
Set PLC Password	660

General Information

Introduction

The PLC and CPU functions can be controlled in ONLINE mode. The PLC must be connected to the host computer to establish ONLINE mode.

You can control the PLC directly with the following commands:

- Set Scan Time
- Single Scan Function
- Clear Controller
- Set Clock
- Run Optimized Solve
- Save in Flash
- Set Password for PLC

The commands for setting up and controlling the PLC can be found in **Online** → **Online Control Panel**.

Setting the Time for Constant Scans

Introduction

A constant cycle time for processing the user program can be specified in the **Online →Online control panel →Invoke constant sweep →Constant Sweep Settings** dialog box.

However, if the actual cycle time is longer than the constant cycle time specified the system ignores the user settings and uses the normal cycle running time (**Cycle time in free running mode**).

If a constant cycle time is selected which is longer than the actual cycle time, the control will wait during each cycle until the set cycle time has been reached.

NOTE: Inputs/outputs connected via communication experts may not be used for updating constant I/O requests, as there can be highly variable I/O response times.

NOTE: This function cannot be performed when there is a link with the simulator.

Selection condition

This dialog box is only available if the link has been established between the PLC and the programming device (ONLINE mode).

Settings for constant cycle

A tab (4x) must be specified first to determine the constant cycle. You also need to enter the scan time (10-200m) that is allocated to the register.

NOTE: The scan time increases if several windows are open in Concept, e.g. several sections are displayed in animation mode. Therefore if you are using several windows you should reduce the scan time.

Exiting Constant Scan

After starting the constant scan with the **Invoke constant sweep...** changes the designation of the command button in **Cancel constant sweep...**. Clicking on this command button exits the function.

Single Sweeps

Introduction

You can specify single sweeps times for processing the user program in the**Online →Online Control Panel →Single Sweep On… →Settings for Single Sweeps** dialog box.

After the specified number of scans has been performed the logic editing stops. This function is helpful for diagnostic purposes. It allows the checking of edited logic, modified data and calculations that have been carried out.

⚠ WARNING

It can lead to unsafe, dangerous and destructive operations of the tools or processes that are attached to the controller.

Single sweeps should not be used for searching for errors in controlling machine tools, processes or material maintenance systems if these are running. When the number of scan times given has been processed, all the outputs will be retained in their last state. Since no more logic editing is taking place, the controller ignores all input information. Therefore the single sweeps function should only be used for searching for errors during start up.

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

Selection Condition

This dialog box is only available if the link has been established between the PLC and the host computer (ONLINE mode). Single sweeps are only performed in PLC RUN mode.

Settings for Single Sweeps

To determine the single sweeps, the scan time (10 – 200ms) and the number of scans being performed must be specified. A maximum of 15 single sweeps is possible.

Execution of Single Sweeps

After the scan time and number have been specified you can perform the single sweeps with the **Trigger Sweep** command button.

NOTE: The **Trigger Sweep** command button is only available in PLC RUN mode.

Exiting Single Sweeps Function

After the single sweeps function has been started with the **Single Sweep On** command button, the designation of the command button changes to **Single Sweeps Off**. Clicking on this command button terminates the function again, and the **Settings...** and **Trigger Sweep** command buttons no longer appear in the dialog box.

Deleting memory zones from the PLC

At a Glance

Specific memory zones can be deleted from the PLC by activating the corresponding options key in the**Online →Online Control →Delete PLC... →Delete PLC contents** dialog box.

The menu command **Load...** can be used to load the deleted memory areas back onto the PLC.

Condition for dial-in

This dialog box is only available if the link has been established between the PLC and the programming device (ONLINE mode) and the PLC is in STOP mode.

Deleting a configuration

If the hardware function of a PLC is deleted, no further online functions can be performed. The NOT CONFIGURED and NOT EQUAL TO modes are displayed in the status bar.

Deleting a program

If the user program is deleted in the PLC, the PLC cannot be started. The NOT EQUAL TO state is displayed in the status bar.

Deleting state RAM

If the state RAM is deleted, the initial values of the located variables in the PLC are set to 0.

Speed optimized LL984-Processing

At a Glance

A speed optimized LL984 Processing can be optimized in the dialog box**Online** → **Online Control** with the **Opt. processing in** command button.

After the command button is activated its designation changes in **Opt. Processing out**. This means that a click on this command button will deactivate the speed optimization which is running again.

NOTE: This function only influences the LL984- program.

Condition for dial-in

This dialog box is only available if the link has been established between the PLC and the programming device (ONLINE mode) and the PLC is in STOP mode.

Save To Flash

Introduction

For data protection purposes, you can save parts of the RAM in the PLC's Flash-EPROM. After a power failure, the contents of the Flash-EPROM is loaded back onto the CPU RAM for the restart.

⚠ **WARNING**

Modified process status after next start!

It is important to choose the right time for saving to Flash, as there could be signal values in the Flash memory which are downloaded later following a power failure, and which do not correspond to the process status for the next start.

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

Selection condition

This function is available when using all 140 CPU 434 12 and 140 CPU 534 14 TSX Compact, Momentum and Quantum modules.

This function is not available with IEC Hot Standby operation with Quantum.

The Flash memory function is not available when using the simulator.

Procedure

Carry out the following steps to Save To Flash:

Step	Action
1	In the **Flash Type** area, select the **Internal** or **PC Card** option button depending on the hardware used. **Note:** Applications that require more than 480 kBytes must be saved in the PC Card Flash.
2	In the **Controller State** area, select the operating mode (RUNNING or STOPPED) the PLC should be in after a restart.
3	Check the **Allow Editing After Power Up** check box if you want to edit the uploaded Flash program when the voltage supply returns. **Caution:** Since these later modifications were not downloaded onto the Flash-EPROM, this data is lost if there is a power failure.
4	Check the **Save State Ram** check box if you want to save all 4x registers to Flash-EPROM. **Note:** This selection is not available with the Momentum family, i.e. all applications are always downloaded to Flash-EPROM.

Step	Action
5	If you have checked the **Save State Ram** check box, you must enter the number of registers to save in the **Number of registers to save** text box. The corresponding register area, which is downloaded onto Flash-EPROM, is then set from the 400001 address.
6	Select the **Save To Flash** command button to load the user program, the configuration, the IEC programming initial values from the RAM to the Flash EPROM.

Edit Flash program

As soon as the **Allow Editing After Power Up** check box is checked, on saving to Flash, information is loaded to the Flash-EPROM, which after uploading the contents of Flash (e.g. in the case of the return of the power supply) allows the program to be edited. Since these later modifications were not downloaded onto the Flash-EPROM, this data is lost if there is a power failure. To prevent this, changes should be downloaded to Flash-EPROM by using the **Save To Flash** command button.

Modification of the Flash program is not allowed.

As soon as the **Allow Editing After Power Up** check box is unchecked, the program can be modified after reading the Flash contents (e.g. in the case of the return of the power supply), but cannot be loaded to the Flash EPROM.

Modifying the program leads to the following reactions when uploading:

Procedure:	Changes protected with Download changes...	Changes protected with Save project	The following status is established after connection:
a)	Yes	No	EQUAL
b)	Yes	Yes	NOT EQUAL

If the EQUAL status is established in the above case a), the contents of the host computer are different from the contents of the Flash-EPROM. After a power failure the Flash-EPROM is uploaded, resulting in the loss of all changes.

If the NOT EQUAL status is established in the above case b), the contents of the Flash-EPROM are different from the contents of the host computer. After a power failure the Flash-EPROM is uploaded, resulting in the loss of all changes.

NOTE: To download a program change to Flash EPROM again, the **Save To Flash** command button must be available again. Specific steps must be carried out to do this, as described in the *Reactivate flash save, page 659* section.

M1 Ethernet CPU

The password protected application is automatically downloaded on each switching on/off cycle. This procedure cannot be undone if you forget the password. The PLC must be sent for repair.

Reactivate flash save

Introduction

If you have not checked the check box in Flash Save **Allow Editing after Power Up** the program saved in Flash EPROM can no longer be changed. After a power failure the Flash-EPROM will finish on restarting the PLC. However, the command buttons **Save to Flash** and **Clear Flash** are not available.

Reactivate Flash Save

In order to enable the Flash Save again, the following steps are necessary:

Step	Action
1	Turn off the controller.
2	Compact CPUs: Set the "Memory Protect" switch (Memory Protect) to ON. Quantum CPUs: Set the switch to the "stop" position.
3	Turn the controller on again.
4	Compact CPUs: Set the "Memory Protect" switch (Memory Protect) to OFF. Quantum CPUs: Set the switch to the "start" position.
5	Make the link between the host computer and the controller (**Online** → **Connect...**).
6	Open the dialog box Save to Flash (**Online** →**Online control panel** →**Flash Program...**). **Result:** The command buttons **Save to Flash** and **Clear Flash** are now available again.

Set PLC Password

Introduction

You can use a password to prevent the PLC being written to without permission.

Before you can set a new password, however, you must first download the configuration to the PLC. Then enter the password that is to loaded to the PLC. The password is now saved so that password protection operates when a connection is made between the host computer and the PLC (password required).

NOTE: When setting a Quantum password, a specific time can also be set for the automatic cancel function in the **Quantum Security Parameter** dialog box. This function is found in the preference setting **Never**. This function means that the user is logged out after the time specified as soon as no read or write access occurs from the programming device to the PLC through this connection within the predefined amount of time.

Valid characters for the PLC password and user name

The following characters are permitted within the character length of 616 characters:

- a ... z
- A ... Z
- 0 ... 9
- _

NOTE: Spaces, umlauts (e.g.: ä, ö, ü) and special characters are not allowed!

Selection conditions

This function is available when using all TSX Compact CPUs, a Quantum CPU 434 12 A/534 14 A/B or a Momentum Ethernet CPU.

Note

The following passwords can be assigned in Concept:

- PLC password
- Concept Password *(see page 767)* (in Concept Security)

Set new PLC password

To set a new PLC password, proceed as follows:

Step	Action
1	Using **Online** →**Download** load the configuration onto the PLC
2	Using **Online** →**Online Control Panel...** →**Set PLC password...** open the dialog **Change PLC Password**.
3	Enter your new password in the **Enter New Password:** text box.
4	Enter the new password in the **Confirm New Password:** text box again.
5	Enter the user name in the **User name** text box, e.g. "anyname".
6	Press the **OK** command button. **Reaction:** The dialog box is closed and the password is automatically downloaded to the PLC

Change Old PLC Password

To change an old PLC password, proceed as follows:

Step	Action
1	Using **Online** →**Online Control Panel** →**Set PLC password...** open the dialog **Change PLC Password**.
2	Enter your old password in the **Enter Old Password:** text box.
3	Enter your new password in the **Enter New Password:** text box.
4	Enter the new password in the **Confirm Password:** text box again.
5	Enter the user name in the **User name** text box.
6	Press the **OK** command button. **Reaction:** The dialog box is closed.
7	Using **Online** →**Download** load the configuration onto the PLC **Reaction:** The password was loaded onto the PLC, and will be requested the next time the PLC and the host computer are connected.

If You Forget Your Password

If the PLC password has been forgotten the procedure depends on the PLC platform used.

Quantum and Compact:

Step	Action
1	Switch off the power supply to the PLC.
2	Move the Memory Protect switch on your hardware module to the MEM_PROT position.
3	Remove the lithium battery from the PLC.
4	Wait 5 minutes and then switch on the power supply to the PLC again. **Reaction:** By doing this, the battery backup RAM is deleted without downloading the PLC program from Flash-EPROM. In this way, the start status of the PLC (configuration-free and without log on password) is re-established.
5	Continue with the step table *Set new PLC password, page 661*.

Momentum without Flash:

Step	Action
1	Switch off the power supply to the PLC.
2	Remove the battery from the interface adapter.
3	Wait 5 minutes and then switch on the power supply to the PLC again.
4	Continue with the step table *Set new PLC password, page 661*.

Momentum with Flash:

Step	Action
1	Switch off the power supply to the PLC.
2	Send the module back to the product manufacturer (Schneider Automation GmbH).

20.4 Selecting Process information (status and memory)

Overview

This chapter contains information about selecting the process information.

What's in this Section?

This section contains the following topics:

Topic	Page
General information	664
PLC state	665
Memory Statistics	667

General information

At a Glance

Certain processes and their storage occupancy can be controlled during operation of the automation equipment.

NOTE: Errors can occur when selecting a configuration that has been generated by another configuration tool (e.g. SyCon, CMD). The selection is based on removing the memory, whereby this is not always compatible with the other software programs. Therefore please use the Modsoft Converter to transfer the Modsoft application according to Concept.

Read status bits.

Status bits provide information about the hardware communication with other modules as well as existing errors in the running of the program. The user specifies the status register already during configuration. In this register , status bits that change their state as soon as a faulty signal is set in the process or a timeout word is not observed are saved. The user can recognize via defined status states (0 or 1) whether the process is faulty.

Read storage occupancy

The user can control the storage occupancy for the current project in the memory statistics. In an overview the total memory, free memory space and used memory for the user program, as well as the user files and FFB libraries are displayed.

PLC state

At a Glance

All the PLC states are displayed in the multi-page dialog box.

There are 67 pages altogether, containing various state information

Condition for dial-in

This function is only available if a link has been established between the PLC and the programming device. When the simulator is active the PLC states cannot be retrieved.

Programming states

The following status information is obtained through the programming:

- Number of segments
- End of logic pointer address
- Run/Download/Debug Status

Hardware states

The following state information is given about the hardware:

- CPU state
- S911 Hot Standby State
- Machine State
- State of the I/O processor
- Quantum I/O state
- DIO-State

Error codes

The following state information is given about errors arising:

- Machine stop code
- Quantum start error code S908

Transfer and communication states

The following state information is given about transfer and communication executions:

- Data transfer state
- Message transfer state
- Communication state

Cable A + B states

The following state information is given about the A + B cable:

- Cable A + B error counter
- A + B global state
- Cable A + B communication error counter

Memory Statistics

At a glance

An overview of the IEC memory data for the open project is provided in the **Memory Statistics** dialog box. The current scan time will be displayed if a real PLC is used (i.e. not the simulator). This dialog box does not show information concerning LL984 memory. Information for IEC HSBY memory, which is part of the state RAM (number of input registers), is not shown either.

Total IEC memory

The value shown for the total memory is the value specified in the **PLC Selection** dialog box.

Modifying the total IEC memory size

The total IEC memory consists of the IEC program memory, the global data, and the EFB memory. Additional space is required in the total IEC memory for program expansions and for the administration of program changes. The general recommendation is to set the value in such a way that 20-30% of the value entered in the **Used** text box will remain free.

NOTE: Changes can only be made offline and are only applied once the program has been loaded onto the PLC.

IEC Program Memory

The program memory includes the program code, the EFB code, program data (section data and DFB instance data), upload information, diagnosis information, and administration information.

Memory information	Meaning
Configured	Once you have defined the size of the total IEC memory, the global data, and the size of the EFB memory, you will obtain information regarding the size of the IEC program memory (IEC program memory = total IEC memory - global data - EFB memory).
Free	Shows the available IEC program memory.

The values displayed correspond to the memory space used for:

- Program code
- EFB code
- Program data (section and DFB instance data)
- Upload information
- Diagnostic information
- Administration information

Defragmentation

The value displayed corresponds to the current Defragmentation status. Defragmentation is enabled in the **PLC Selection** dialog box. Defragmentation is a continuous process that ends with a value of 0 after a certain period. "0" in this case means that there are no more gaps in the PLC's memory.

Since this continuous process affects the scan time, it can be disabled in the **PLC Selection** dialog box.

Global Data

The memory statistics cover the following information:

Memory information	Meaning
Configured	The value shown corresponds to the memory space set for unlocated variables in the **PLC Selection** dialog box.
Free	The available memory for unlocated variables is shown here.

Changing the memory space used for global data

You can change the memory space used for global data. It should be noted that increasing the memory space for global data will decrease the IEC program memory size. Each object, e.g. FFB instance, variable, step etc., takes up several bytes in the IEC program memory.

Since deleting unlocated variables does not free up more memory space automatically, we recommend making plans so as to have sufficient memory space. The general recommendation is to set the value in such a way that 20-30% of the value entered in the **Used** text box will remain free.

NOTE: Changes can only be made offline and are only applied once the program has been loaded onto the PLC.

EFB Memory

The EFB memory is used by the user program that contains the EFB code. The latter must not be moved during the memory defragmentation process:

Memory information	Meaning
Configured	The EFB memory size configured in the **PLC Selection** dialog box for **PLC memory defragmentation** is shown here.
Free	Shows the available EFB memory.

Scan Time

The value displayed corresponds to the current scan time. With the first call, the I/O station is standardized so that a scan time of 0 ms/scan is shown. After initialization, the scan time is calculated as an average value.

NOTE: If you are using the simulator, the scan time is not shown. **na** means “not available”.

20.5 Loading a project

Overview

This chapter contains information about loading a project.

What's in this Section?

This section contains the following topics:

Topic	Page
General information	671
Loading	672
Download Changes	674
Uploading the PLC	677
Upload Procedure	679

General information

At a Glance

To carry out an online command a transfer has to be made to the PLC after setting up or changing sections. Otherwise a complete project can be transferred from the PLC to the programming device. As soon as the user program is consistent on the programming device and the PLC, the EQUALS status is displayed in the status bar.

The status display MODIFIED identifies the program in which at least one section has been changed or where changes to the variable editor were performed. With command button **Load changes...** the consistency between the programming device and the PLC is restored. Status display NOT EQUALS identifies a program in which critical changes were performed. Critical changes are for example changes to EFBs, DFBs or derived data types. With command button **Load...** the consistency between the programming device and the PLC is restored.

Loading, loading changes and selecting are not possible in the animation mode.

With command button **Select...** the following project areas can be selected from the PLC:

- Configuration
- IEC sections
- 984 Ladder Logic sections
- ASCII messages
- State RAM
- Initial values
- Extended memory

Process for loading

Loading the PLC can take place in two parts:

1. The exportable code (machine code) is always loaded onto the PLC.
2. The complete compressed user program is loaded onto the PLC

NOTE: The user program, consisting of user defined EFBs, DFBs, derived data types and the program (variables, sections, etc.), is only loaded onto the PLC if in dialog **Options for generating codes** (**Project →Options for generating codes...**) check box **contain IEC selection information** was activated beforehand. The option to also load the comments contained in the check box onto the PLC, thereby making them available as selection information, is available, as well.

During selection the entire user program can be transferred from the empty project to the programming device.

Loading

Introduction

With menu command **Load...**the configuration, the entire user program (IEC or LL984 sections) ASCII messages (only with Concept for Quantum) and the state RAM with the initial values of a project can be sent to the PLC. This establishes consistency between the user program on the programming device and the PLC so the online functions are executable.

Loading single parts onto the PLC

Single parts to be loaded onto the PLC can be selected.

The following table contains the available options and their meaning:

Option to be loaded	Meaning
Configuration	This option sends the hardware configuration to the PLC. **Note:** The Hardware Configuration can only be sent to the PLC when a corresponding access privilegehas been authorized. This option is not available with a Modbus Plus connection.
IEC Sections	This option sends the code from all the sections created with an IEC programming language (FBD, SFC, LD, IL, ST) to the PLC.
984 Ladder Logic	This option sends the code from all the sections created with an LL984 programming language to the PLC.
ASCII messages	This option sends ASCII messages for Ladder Logic to the PLC. **Note:** This function is only available when using Concept for Quantum.
State RAM + Initial Values	With this option, at first all initial values of the Located 4x-Variables are copied from the Variable Editor into the state RAM mirror (image). Then, the initial values and all blocked 0x- and 1x-I/O bits from the state RAM mirror (Image) are loaded into the PLC. **Note:** While the PLC is running, all non-blocked 0x-variables are reset by the firmware in the PLC. Thus, values of 0x- and 1x-variables are only loaded if the variables are in blocked status.
Only state RAM	With this option, the initial values of the Located 4x-variables and all blocked 0x and 1x I/O bits are loaded from the state RAM mirror (Image) into the PLC.
Only initial values	With this option, exclusively initial values of the Located 4x-Variables are loaded from the Variable Editor into the state RAM.
Expanded memory	This option assigns the PLC an extended memory allocation (6x-references). **Note:** This function is only available when using Concept for Quantum.

Loading IEC selection information

To obtain a complete project when uploading from the PLC **Options for code generation** the check box **Include IEC Upload Information** must be activated in the dialog box before loading. If this check box is not activated only the executable code (machine code) is loaded onto the PLC.

If loading is not possible

There are several possibilities why loading is not possible:

- An active screen saver can lead to loading errors. It is therefore recommended to deactivate the screen saver.
- If loading the program is not possible due to insufficient program data memory, the memory size can be optimized.*Main structure of PLC Memory and optimization of memory, page 155*.

NOTE: If, while loading the program, a warning appears due to inconsistent DFB versions, use the menu command **Project** →**Synchronize nested DFB versions**.

Download Changes

Introduction

Download changes is always used if sections have been changed, added or deleted, whether online or offline, and the program is in MODIFIED state. In this way the changes are indicated and can be transferred to the PLC.

The changes are loaded into the PLC and the consistency between the user program on the programming device and the PLC is restored.

With changes that do not influence the logic of the program (e.g. renaming a step name, renaming a section, renaming a variable, graphic moving of a component, etc.), the state of the program between PLC and host computer will remain EQUAL and cannot be downloaded to the PLC using the **Download changes** function. However, the changes will not be lost. They can be updated with the **Download changes** function during the change, which will result in the state of CHANGED. Or the entire project is downloaded to the PLC using the **Online** →**Download** function.

NOTE: However, if you wish to update your changes immediately, simulate a Modify code (e.g. delete and restore), so that the state of the program changes to MODIFIED. Then carry out the **Download changes** function.

If the changes cannot be downloaded because there is too little memory in the PLC, there are 2 possibilities for proceeding:

- Sequential loading of modified sections
- Optimize Project

NOTE: If, while loading the program, a warning appears due to inconsistent DFB versions, use the menu command **Project** →**Synchronize nested DFB versions**.

ID for specific sections

The following sections contain additional ID information as they are different from cyclically set sections:

- **E** for "**E**vent Section" (I/O Event and Timer Event Section = Interrupt-Section)
- **T** for "**T**ransition Section"

Sequential loading of modified/new sections

The user can download changed/new segments onto the PLC one after the other.

When segments are downloaded sequentially, the following points must be noted:

- If the constants value has been changed, it is not possible to download the changed segments sequentially.
- All deleted IEC segments will be automatically deleted the first time the user downloads sequentially onto the PLC.
- All initial values of new variables, all modified values of literals are automatically loaded onto the PLC on the first sequential loading.
- If new sections already contain used variables, the value of these variables remains.
- When closing a project ensure that it is saved before loading changes onto the PLC. Otherwise it might not be possible to continue the project after it is reopened with the remaining changes loaded, or there will be "newer" sections (previously loaded changes) on the PLC than on the programming device.

⚠ CAUTION

Risk of unwanted and dangerous process states

Loading sections sequentially on a running PLC can lead to unwanted and dangerous process states. It is therefore recommended to stop the PLC during sequential loading.

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

Modified Initial values

Modified initial values are no longer loaded onto the PLC. The initial value during the first download (**Download...**/**Download changes...**) that was downloaded to the PLC cannot be overwritten with the menu command **Download changes...**. The initial values can however be changed in the Reference data editor.

Procedure for sequential loading

The procedure for downloading changes sequentially is as follows:

Step	Action
1	Stop the PLC with **Online** →**Online control** →**Stop controller**.
2	Select the segment(s) which must be downloaded from the list.
3	Confirm using **OK**.
4	Call the dialog box again and repeat the process until all modified/new sections are loaded onto the PLC and **EQUAL** mode is reached.
5	Start the PLC with **Online** →**Online PLC** →**Start PLC**.

Loading IEC upload information

If, in the **Code generation options** dialog, the check box **Include IEC upload information** is checked, IEC upload information is loaded onto the PLC with the **Download changes...**menu command.

Optimize the Project

It may be possible to eliminate existing gaps in the program data memory management of the PLC with the menu command **Optimize project**and enable loading again in this way. However, the PLC must be stopped and the complete program must be downloaded again. Furthermore, it may be necessary to adjust the size of program data memory, see Memory statistics *(see page 667)*.

It is still possible to optimize use of the program data memory with the menu command **Online** →**Memory statistics**.

⚠ CAUTION
Modifications are only transferred when the program is loaded onto the PLC.
After optimizing the project or modifying the program data memory size the PLC must be stopped and the program loaded onto the PLC.
Failure to follow these instructions can result in injury or equipment damage.

Uploading the PLC

Introduction

With menu command **Upload...**the configuration, the entire user program (IEC and/or LL984 sections), the ASCII messages and the state RAM with a project's initial values can be transferred from the PLC to the host computer.

NOTE: Upload information (PLC configuration), which was generated by the Software programs as Concept, is possibly erroneous. The upload is based on removing the memory, so that it is not always compatible with other software programs.   Please use the Modsoft converter for transferring your Modsoft application to Concept.

Reading Individual Parts from the PLC

Individual parts to be loaded from the PLC to the host computer can be selected.

The following table contains the available options and their meaning:

Option to be loaded	Meaning
Configuration	This option sends the hardware configuration to the host computer. **Note:** The hardware configuration can only be sent from the PLC when a relevant authorization is granted in the Access Rights. This option is not available with a Modbus Plus connection.
IEC sections	This option transfers the revertive presentation information of all the sections created with an IEC programming language (FBD, SFC, LD, IL, ST) to the host computer. In this process, however, no current signal values from variables and registers are loaded.
984 Ladder Logic	This option sends the revertive information from all the sections created with an LL984 programming language to the host computer.
ASCII messages	This option transfers ASCII messages for Ladder Logic to the host computer. **Note:** This function is only available when using Concept for Quantum.
Upload state RAM + update initial values	With this option, first all Located 0x-, 1x- and 4x-values are read from the SPS, and saved in the state RAM mirror (Image). Then, the initial values of the 4x-variables are overwritten with the value from the state RAM mirror (Image). With the upload process, the dialog box **Upload initial values** is then opened. With the command button **Yes** you confirm the overwriting of the initial value displayed with the new value. **Note:** Uploaded state RAM values can be overwritten in the RDE by Online Operations. This behavior can be changed, however, in the CONCEPT.INI-file *(see page 1118)*.

Option to be loaded	Meaning
Only update initial values	With this option, the initial values of the Located 4x-Variables are overwritten by the Variable Editor with values from the state RAM.
Only upload State RAM	With this option, all Located 0x-, 1x- and 4x-values are read from the SPS, and saved in the state RAM mirror (Image). The initial values in the Variable Editor are not overwritten.
Expanded memory	This option transfers the PLC's available extended memory (6x references) into the configuration. **Note:** This function is only available when using Concept for Quantum.

Upload Procedure

Introduction

If the IEC upload information was being taken into account during loading into the PLC (**Project →Code Generation Options →Include IEC upload information**), a new project containing the IEC upload information is generated in Concept during upload. In this way, the entire user program and user EFB libraries are always downloaded, i.e. individual sections, EFBs etc, cannot be selected for transfer.

NOTE: During loading (**Online →Download Controller**) of the IEC upload information, additional memory is required so that this function should only be used, when you want to upload the project loaded into the PLC again.

Requirements

In order to carry out a PLC upload, an empty project must first be created.

There are several ways of doing this.

Selection	Action
1	You can create an empty project using the **File →New project** menu command. Then execute the **Online →Upload...** menu command. **Result:** The **Upload to project** dialog is opened. Here you can determine (e.g. D:\NEW\TESTPRJ.PRJ) where the project will be uploaded to. **Note:** You can select a different directory or even create a new directory so as not to come into conflict with existing projects. The preset project name corresponds to the project name downloaded in the PLC and does not necessarily have to be changed.
2	Using the **File →Open...** menu command you can create a new project (e.g. D:\NEW\TESTPRJ.PRJ) Then execute the **Online →Upload...** menu command. **Result:** The **Upload Controller** dialog is opened.
3	There is **no** project open and you have established a connection with the PLC using the **Online →Connect...** menu command. Then execute the **Online → Upload...** menu command. **Result:** The **Upload to project** dialog is opened. Here you can determine (e.g. D:\NEW\TESTPRJ.PRJ) where the project will be uploaded to. **Note:** You can select a different directory or even create a new directory so as not to come into conflict with existing projects. The preset project name corresponds to the project name downloaded in the PLC and does not necessarily have to be changed.

Procedure

To upload loaded IEC information, proceed as follows:

Step	Action
1	Open a new project. **Note:**If, during upload, there is a second project still open, it must be closed. In this case a query appears asking whether the project should be saved before it is closed and all changes are lost.
2	Establish a connection between the PLC and the programming unit (**Online** → **Connect...**).
3	Start the upload procedure (**Online** →**Upload Controller...**). **Result:** A window appears in which you can determine the path for the project that is to be uploaded.

Double Designation

Conflicts with existing names can occur during the upload procedure.

Double designation is prevented for each program sequence as follows:

Program sequence	process
User EFB library	A query appears, which can interrupt uploading. If not, a query appears, asking whether the user EFB library should be overwritten, and whether a backup of the old EFB library should therefore be created.
DTY File (derived data types)	A query appears, which can interrupt uploading. If not, the DTY file of the same name is automatically overwritten. No backup is made of the old file.
DFB library	A query appears, which can interrupt uploading. If not, the DFB file of the same name is automatically overwritten. No backup is made of the old file.

20.6 Section animation

Overview

This chapter describes the basic principles for animating sections. The details can be found in the chapters on individual programming languages.

What's in this Section?

This section contains the following topics:

Topic	Page
IEC-Sections animation	682
LL984 Programming Modes	684

IEC-Sections animation

At a Glance

IEC sections can be animated, i.e. the program's current states in the PLC /simulator will be displayed.

Animation is possible with both a running and a stationary PLC. Display data is automatically refreshed when the PLC is running. The static state of the program on the PLC is displayed when the PLC is stationary.

Load and **Load changes** is not possible in animations mode. Should these commands be executed, animation will be stopped automatically.

Requirements for animation

Requirements for animation

- The section to be animated in the programming device and the section loaded onto the PLC must be consitent. Otherwise, establish consistency using **Online** →**Load...** (if mode **UNEQUAL**) or **Online** →**Load changes...** (if mode **MODIFIED**).
  Note: Even when the program mode is **MODIFIED**, the sections that have not been changed can be animated. The mode displayed in the footer refers to the program and not to the currently displayed program.
- To animate, the programming device and the PLC must be online. Otherwise, establish the link using **Online** →**Link...** .

Active animation display

The active animations mode is indicated:

- by a check mark before the menu command, in the **ANIMATED**box on the status bar,
- by a depressed animations button on the symbol bar and
- by the gray window background.

Animating more than one section

If several sections are animated, an animated section is updated in each cycle. This means that the more animations are active, the "older" the values of the individual animations. Additionally, the animation increases the load on the PLC cycle. For this reason, animations that are no longer necessary should be terminated. This also applies to the animation of many variables or very large derived data types.

NOTE: When coupling using Modbus Plus, it is recommended that no more than 10 sections should be animated at one time.

NOTE: When coupling using Modbus, it is recommended that no more than 5 sections should be animated at one time.

Animating a disabled section

If a disabled section is animated, the state **INHIBITED** is displayed in the status bar.

Animation of a transition section

If the animated section is used as a transition section for the sequential control (SFC), and the transition (and therefore also the transition section) is not processed, the status **INHIBITED**appears in the transition section.

Changing a animated section into a symbol

If an animated section is changed into a symbol, the animation with the most current values stops, and then restarts automatically once the section is called.

LL984 Programming Modes

Direct Programming

There are two situations that determine how direct mode ladder editing is applied. The first is where there is no open project and you are connected to a PLC that has a valid program in it. When you select the command **Direct Mode LL Editor** the first program in the first segment is displayed. You can see the direct mode status at the right side of the status bar and the network window is labled **984 LL Direct**.

The second case occurs when you have a project open and you are connected to the PLC (but not **EQUAL**). When you select **Direct Mode LL Editor** in this case a dialog is displayed listing segments and the number of networks contained in each. Click on the segment you want click on **OK** and the network edit window is displayed with a window labeled **984 LL Direct**. If you have an orignal edit window it will remain on the display.

Combination Mode

Combination programming occurs when the programming panel is online. Valid program changes are immediately written to both the controller and the program database simultaneously.

20.7 Online Diagnosis

Diagnostics Viewer

Introduction

Using the diagnostics viewer in Concept (**Online →Online Diagnostics...**) it is possible to view the content of the PLC diagnostics error buffer.

Selection Condition

The diagnostics viewer is only available if the PLC is in online mode and the EQUAL status has been created between the PLC and host computer.

The diagnostics viewer only works with the SFC, FBD and LD programming languages and with the diagnostics blocks of the EXTENDED group.

Conditions of the Diagnostics Viewer

To activate diagnostics, a supervision time must first be set for the step (Transition diagnostics) or the diagnostics block (Reaction diagnostics). In addition, in the **Code generation options** dialog (**Project →Code generation options...**), the **Include diagnosis information** check box must be checked. As a result memory space is prepared on the PLC (max. 64 diagnostics entries) for the diagnostics error buffer.

Behavior of the error buffer

A maximum of 64 events (errors) and a maximum 20 signals per event are read. If the diagnostics error buffer overflows all further signals (21 onwards) are lost. The next event (error) coming is only entered once an event (error) which has gone has been acknowledged in the error buffer.

A diagnostics error buffer overflow is displayed in the dialog status line.

NOTE: A maximum of 16 events (errors) can be scheduled within one SFC section. All further errors (17 onwards) are lost. The next event (error) is only entered once a past event (error) has been acknowledged in the error buffer.

Transition Diagnosis

Information on this can be found in the *Transition diagnosis, page 319*section.

Reaction diagnosis

Information on this can be found in the "Diagnostics Block Library" handbook.

Diagnostics viewer

After analysis, the events (errors) and the analyzed signals are written in the buffer and displayed in the diagnostics viewer in Concept.

The following specific information is contained in transition diagnostics:

- Denotes the transition preventing the active step from being executed to the next step.
- Denotes the TRANS type for transition in a PLC section
- Denotes the active step, which is not executed.
- If this is a transition section in the named transition, the analyzed signals are also listed.

The following specific information is contained in reaction diagnostics:

- Denotes the diagnostics block preventing a reaction from being triggered due to incorrect signals.
- Denotes type ACT, PRE, GRP, LOCK, REA for diagnostics blocks
- Diagnostics block drop number
- The analyzed signals are listed.

20.8 Logging Write Access to the PLC

Logging and Encrypted Logging

Introduction

Logging the write access to the PLC can record the following data among others:

- Section name
- EFB/DFB Instance name, FB type name
- Pin-Name
- [variable name] [literal] [address]
- Old value
- New value
- User name (if the Concept (Login) password is activated in Concept Security)
- Date and Time (see also*Address format in LOG file [Logging], page 1116*)

The following logging can be carried out during log-on:

- Modification of the user rights
- Deleted user
- Aborted log-on

Besides the log which can be read in the *.LOG file, an encrypted log can be created in an *.ENC file. The file name is made up of the current date, e.g. 20020723.LOG or 20020723.ENC.

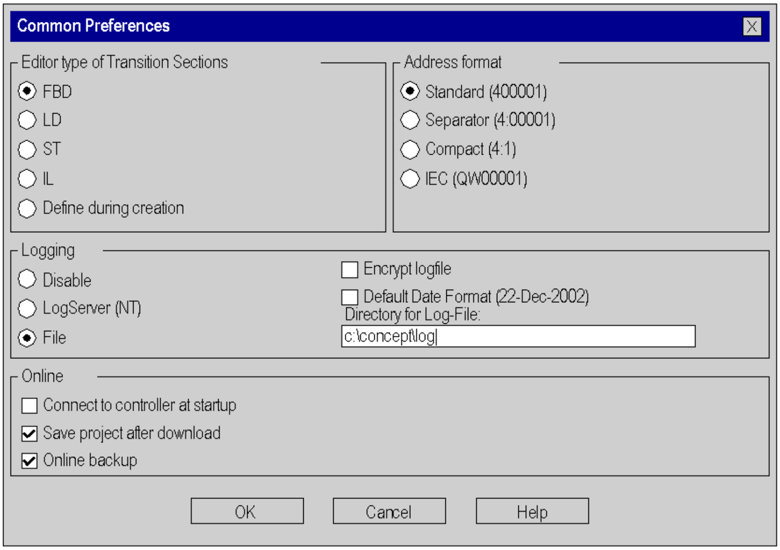
Encrypting the protocol file is done to protect the file contents from being changed. The view tool is only provided so that the user can read the log file. Saving the file in another readable format is not possible. Editing the file so that it is not recognizable is impossible since the ASCII file only displays unrecognizable characters.

NOTE: Log files are not archived by Concept and no backup files are created.

Log *.LOG

Logging is activated in Concept using the **Options** →**Preferences** →**Common...** → **Common preferences** with the **File** option dialog box. Use the text box **Directory for Log-File:** to define a new path for the log file (e.g. 20020723.LOG).

Dialog **Common Preferences**:



Common Preferences

Editor type of Transition Sections

- ● FBD
- ○ LD
- ○ ST
- ○ IL
- ○ Define during creation

Address format

- ● Standard (400001)
- ○ Separator (4:00001)
- ○ Compact (4:1)
- ○ IEC (QW00001)

Logging

- ○ Disable
- ○ LogServer (NT)
- ● File
- □ Encrypt logfile
- □ Default Date Format (22-Dec-2002)

Directory for Log-File: c:\concept\log

Online

- □ Connect to controller at startup
- ☑ Save project after download
- ☑ Online backup

The current logfile can be viewed in Concept with menu command **File →View Logfile**.

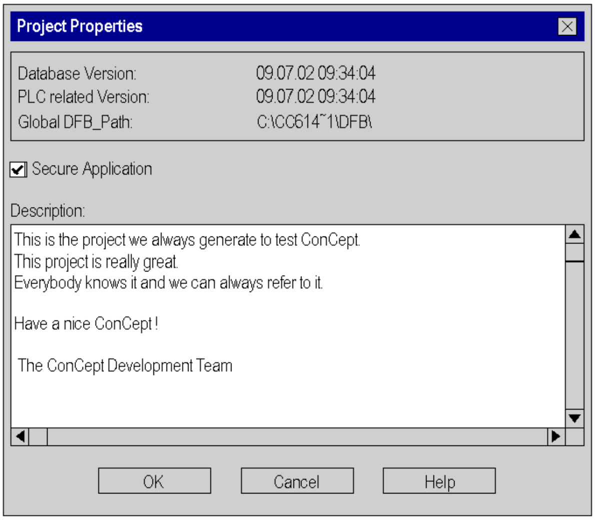
Encrypted Logfiles *.ENC

In addition, any repetitive strings are displayed in separate encrypted strings in the logfile.

In Concept, the encryption can be activated with two different settings:

- With menu command **Options →Preferences →Common →Common Preferences** and the activation of the check box **Encypt Logfile**.
  Note: The check box is only available if **no** project is open.
- Indirectly with the menu command **Project →Project Properties** and the activation of the check box **Secure Application**.
  Note: This dialog box is only available in offline mode.

Dialog **Project Properties**:



Database Version: 09.07.02 09:34:04
PLC related Version: 09.07.02 09:34:04
Global DFB_Path: C:\CC614~1\DFB\

☑ Secure Application

Description:

This is the project we always generate to test ConCept.
This project is really great.
Everybody knows it and we can always refer to it.

Have a nice ConCept !

The ConCept Development Team

If the encryption is activated after an unencrypted logfile (*.LOG) has been created, then a second encrypted logfile (*.ENC) is created. The storage location for the *.ENC file is defined in the **Common Preferences** (**Directory for Log-File:**) dialog box.

NOTE: Supervisor rights are required for activating the encrypted logging procedure.

View Tool

The View tool can be used for reading encrypted logfiles. Editing and saving so that the file can be read normally is not possible but the logfile can be printed. Supervisor rights are required in this case as well. With menu command **File →View Protocol** the View tool is opened automatically if encryption has been activated for the current log.

The logfile is stamped with an electronic signature and the following tests are performed:

- The logfile is created by Concept.
- The logfile is not a counterfeit.

Import/Export

21

Overview

This chapter describes the various import and export options for sections, variables and PLC configurations.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
21.1	General Information about Import/Export	692
21.2	Exporting sections	694
21.3	Exporting variables and derived data types	697
21.4	Section import	698
21.5	Variables import	716
21.6	Import/Export of PLC Configuration	726

21.1 General Information about Import/Export

General Information about Import/Export

Export functions

The following export options are available:

Program	Path	Export files
Concept Concept DFB	**File →Export**	● Sections from a source project and import into a target project ● Sections from a source DFB and import into a target DFB ● Sections from a source DFB and import into a target project ● Sections from a source project and import into a target DFB ● FBD, SFC and LD sections into IL or ST files ● Variable declarations into an ASCII file (Concept only) ● PLC configuration (Concept only)
Concept	**Edit →Save as text file...**	● Contents of IL or ST sections into an ASCII file ● Definitions of Derived data types from the Data type editor
Concept	**File →Archiving...**	Relevant project files (compressed)
Concept Converter	**File →Export → Configuration**	PLC Configuration

Import functions

The following import options are available:

Program	**Path**	**Import files**
Concept Concept DFB	**File →Import**	● Exported sections from a source project or source DFB ● Exported or externally created IL/ST files into IL/ST sections ● Exported or externally created IL/ST files into FBD/SFC sections (with conversion) ● Variable declarations from an ASCII file (Concept only) ● PLC configuration exported with Concept (Concept only)
Concept	**Edit →Insert text file...**	● Contents of ASCII files IL or ST sections ● Definitions of Derived data types into the Data type editor
Concept	**File →Archiving...**	Relevant project files (decompressed)
Concept Converter	**File →Import**	PLC Configuration

21.2 Exporting sections

Exporting Sections

Introduction

In Concept it is possible to export projects or DFBs selectively from a source project/source DFB, and if desired, to then import them immediately into the current target project.

Requirements

The project from which the export is to take place must be stable (check using **Project →Analyze Program**).

NOTE: When exporting IL and ST sections, ensure that the settings for nested comments (**Options →Preferences →IEC Extensions →Allow nested comments**) are identical in the source and target projects.

Export range

The following are exported:

- the selected sections with their accompanying variables, DFBs, EFBs and data types.
- In the case of SFC, the accompanying transition sections are also exported.
- The PLC configuration is **not** exported.

Exporting more than one section

When more than one section is exported, a "pseudo SFC" is generated to maintain the execution order. To do this, the following code is generated:

```
INITIAL_STEP   SECTION_SCHEDULER:
    Section1 (N);
    Section2 (N);
       :
    SectionN (N);
END_STEP
```

Exporting FBD, SFC and LD Sections

Using **File →Export →Program: IEC Text** FBD, SFC and LD sections can be exported to IL and ST. The text languages of both export files follow the grammar of IEC text languages, shown in IEC 1131-3 and in the process tables 52 - 56 of IEC 1131-3.

The exported code is displayed in a PROGRAM … END_PROGRAM or FUNCTION_BLOCK ... END_FUNCTION_BLOCK frame and contains all project or DFB variables in a VAR ... END_VAR frame at the start of the file.

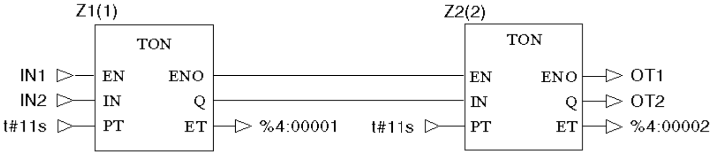
If more than one section is being exported, the code separation is expressed as an artificial PLC frame, which is not a component of the original program. It only has one INITIAL_STEP for all sections linked to it as actions (with the identifier N). These actions (sections) are executed every time the step is active, which is always the case. The actions follow as sections, which do not have variable declarations.

The artificial INITIAL_STEP has the name SECTION_SCHEDULER. It displays the execution order as it was specified in the section execution order dialog box. The artificial SFC frame is left out when re-importing in Concept. The criterion for this omission is the specific name SECTION_SCHEDULER.

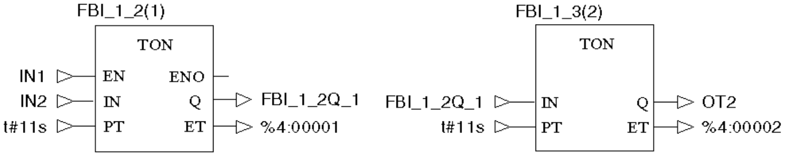
The ASCII file can be re-imported into a FBD or SFC section using the IEC text import. Using exports and imports it is possible, for example, to convert a LD section into a FBD section. Importing into a LD section is not possible.

If the EN and ENO optional imports/exports have been used in the FBD/LD sections, they are ignored when exporting to IL/ST.

FBD section logic before export:



FBD section logic after import:



The LD elements "N.C. contact" and "N.O. contact" are converted to AND and ANDNOT.

The ASCII file can, however, also be imported into an IL or ST section using the **Insert Text File** function. In this case, however, manual correction of the files is necessary, since the extensions described above must be removed again from the file.

SFC Export Limitations

The following limitations should be noted when using SFC import:

- Only variables are permitted as actions. Direct addresses cannot be exported.
- Only literals are allowed as time variables for identifiers. Variables are converted to literals with the value 0.
- Transition section names are changed to standard names.
- Step monitoring times and step delay times are lost when exporting.

Exporting IL and ST Sections

Using **Process →Save as Text File...** it is possible to export the contents of IL or ST sections into an ASCII file.

This export function is a simple text export function, which can also be performed via the clipboard (cut/copy/paste). Data conversion does not take place. For this reason, the required variable declarations, for example, are not exported with the section contents. If the ASCII files are to be converted to an FBD or SFC section using **File →Import →Program: IEC Text**, all information necessary for the project (e.g. program frame, section name (see also *Importing (insert file) IL and ST programs into IL or ST sections, page 712* and *Procedure for "Copying" an IL section from an existing project into a new project., page 713*)) must be entered manually.

21.3 Exporting variables and derived data types

Exporting variables and Derived Data Types

Exporting variables in "Text delimited" format

Using **File** →**Export** →**Variables: Text delimited** a project’s variables can be exported into a ASCII file in text delimited format (refer to *Importing Variables in "Text Delimited" Format, page 717* and *Importing structured variables, page 720*).

The ASCII file can be re-imported into a Concept project with the help of the importing text delimited (refer to *Importing Variables in "Text Delimited" Format, page 717*).

Exporting variables for Factory Link

Using **File** →**Export** →**Variables: Factory Link** a project’s variable declarations can be exported into a ASCII file in "Factory Link" format.

If your Factory Link version of Concept is not supported, please call our hotline.

The ASCII file can be re-imported into a Concept project with the help of Factory Link import *(see page 724)*.

Exporting variables for Modlink

Using **File** →**Export** →**Variables: Modlink** a Modlink configuration file can be generated, which can be used directly in Modlink.

The Modlink configuration file contains all those Located variables, which are selected to be exported in the Variable Editor.

If no Located variables are selected to be exported, an error message appears and a configuration file will not be generated.

Related information about Modlink is found in *Modicon ModLink, User Guide*.

Exporting Derived Data Types

In the data type editor, using **Process** →**Save as text file...** definitions of Derived Data Types can be exported to a ASCII file.

21.4 Section import

Overview

This section describes the import of sections.

What's in this Section?

This section contains the following topics:

Topic	Page
Importing Sections	699
Procedure for importing sections	703
Importing IL and ST Programs to FBD, SFC, IL or ST Sections (with Conversion)	709
Importing (insert file) IL and ST programs into IL or ST sections	712
Procedure for "Copying" an IL section from an existing project into a new project.	713
Procedure for converting FBD sections from an existing project into IL sections of a new project.	714

Importing Sections

Introduction

In Concept, the possibility exists to export individual sections selectively from a source project or source DFB, and if desired, to then import them immediately into the current target project or DFB:

- Section export from source project, followed by section import into the target project
  This transfers section information, including transition sections at SFC, all used global and local DFBs used, as well as all the variable declarations used.
  Data types defined in data type files are not transferred, (refer to notes).
- Section export from source DFB, followed by section import into the target DFB
  This transfers section information, all global and local DFBs used as well as all declarations of variables, inputs and outputs used.
  Data types defined in data type files are not transferred, (refer to notes).
- Section export from source project, followed by section import into the target DFB
  This transfers section information, all global and local DFBs used as well as all used declarations of unlocated variables.
  Direct address and Located variable declarations must be deleted before export, since they are not authorized in a DFB. Data types defined in data type files are not transferred, (refer to notes).
- Section export from source DFB, followed by section import into the target project
  This transfers section information, all global and local DFBs used as well as all declarations of variables used.
  Declarations of inputs/outputs in this DFB must be deleted before export, since they are not authorized in a Concept project. Data types defined in data type files are not transferred, (refer to notes).

Notes

Attention should be paid to the following notes:

- The imported sections will be inserted at the end of existing sections.
- The PLC configuration is not automatically imported. Instead, it must be imported explicitly (refer to *Import/Export of PLC Configuration using Concept, page 727*.
- If projects with different local data structures are being imported (different DTY files in the local DFB directories), they must be brought together in an individual DTY file before import. This common file must then be saved in the local DFB directories for source and target projects. Afterwards, open these files to make them known to the individual projects.
- Ensure during import of IL and ST sections that the settings for nested comments (**Options →Preferences →IEC extensions →Nested comments authorised**) are identical in the source and target projects.

Checking the Sections to be Imported

Before the import actually takes place, a check of the following takes place:

- identical project environment (DFBs, EFBs, definition of Derived Data Types)
- existing sections
- existing SFC sections (not authorized in Concept DFB)
- existing step names
- Declarations of inputs/outputs (not authorized in Concept projects)
- Declarations of direct addresses (not authorized in Concept DFB)

If an error is identified, the import is canceled.

Errors that occur subsequently are "irreparable" and will cause the project to close (i.e. all changes made since the last save are lost). Possible errors are:

- Name collisions between variables with different data types
- Name collisions between item names
- other errors

Name collisions between variables with different initial values or direct addresses (located variable) cause a warning. The value of the target project is retained.

Automatic adjustment of standard preset names

An automatic adjustment of standard preset names occurs in the case of:

- Standard generated names, such as SFC step names (S_x_y) and transition section names (TransSection_x_y)
- Standard generated item names (FBI_x_y)
- Position of new DFB inputs/outputs (only with import into Concept DFB)

Specific Changes

During import there are also the following possibilities for performing specific changes, in order to adjust the sections that are to be imported specifically to the target project / target DFB.

- Replacing names (variable names, section names, item names, names in text languages, comments, etc)
- Address offset for located variables and direct addresses in graphic languages (e.g. %3:10 -> %3:20) and text languages (%QW10 -> %QW20)

The following points are excluded from the replace function:

- DFB names
- Index of ARRAYs (e.g. a[1])
- Elements from multi-element variables (e.g. a.dummy)
- In the case of EFBs, the replace function is only used for non-automatically generated names (i.e. Instance names)

Syntax for replacing names and address offset (address shift)

The following syntax applies when replacing names:

- Only entire names will be searched. If parts of names are to be replaced, wildcards must be used.
- The "?" character is permitted as a wildcard. It is used to represent one character exactly. If more than one character is to be ignored, a corresponding number of "?" must be used. The "?" character is only permitted at the start of the name.
- The "*" character is permitted as a wildcard. It is used to represent any number of characters. The "*"character is only permitted in the character string that is to be searched for.
- Wildcards are only permitted in the search character string.
- There are no case distinctions.
- The section name, which is to be used as a replacement, must conform to IEC name conventions, otherwise an error message occurs.
- In accordance with IEC1131-3, only letters are permitted as the first character of item names. Should figures be required as the first character, however, the menu command **Options →Preferences →IEC extensions... →Allow leading digits in identifiers**.
- The specified value for the address offset is added to the corresponding address zone for located variables and direct addresses.
- The offset value is given in decimal format by default. If it is given in hexadecimal format, this can be marked as such with the prefix "16#" in front of the actual offset value (e.g. 16#100).

NOTE: Replacing names has an effect on all variables, instance names and comments. Using wildcards runs the risk of replacing names that also happen incidentally to contain the same character string that is being searched. This would normally lead to a cancellation.

Examples of search and replace:

Replaces:	By:	available names	Result
Name1	Name2	Name1 Name1A NameA NameB	Name2 Name1A NameA NameB
???123	456	abc123 cde123 abcd123 abc1234	abc456 cde456 abcd123 abc1234
Name1*	Name2	Name1A XName1B NameAB	Name2A XName1B NameAB

Replaces:	By:	available names	Result
*123	456	abc123 cde123 abcde123 abd123a	abc456 cde456 abcde456 abd123a
123	456	abc123abc cde123defghi abcde123def	abc456abc cde456defghi abcde456def
???123*	456	abc123abc cde123defghi abcde123def	abc456abc cde456defghi abcde123def

Syntax for Creating the Replace List with an External Editor

When creating the replace list using an external editor, the following syntax should also be noted:

- The replace-by string (previous name-new name) must be separated by a comma (e.g. name1,name2).
- The replace list is processed line by line. Individual replace instructions must be separated by a line break.
- The instructions for the address offset have the following structure:
  - to add an address offset:
    ```
    <reg0>,www
    <reg1>,xxx
    <reg3>,yyy
    <reg4>,zzz
    ```
  - to subtract an address offset:
    ```
    <reg0>,-www
    <reg1>,-xxx
    <reg3>,-yyy
    <reg4>,-zzz
    ```
  - Likewise, the value can be given in hexadecimal form, e.g.:
    ```
    <reg1>,16#xxx
    ```

Procedure for importing sections

At a Glance

In principle, sections must firstly be exported from the source project /DFB into an export file (*.sec) and then be imported by the target project/DFB. Exporting and importing from project to project or from DFB to DFB can take place in shared or in separate sessions. Exporting and importing from projects into DFBs or from DFBs into projects must take place in separate sessions.

Section export and section import

To section export a source project and then section import into a target project, the following procedure should be performed:

Step	Action
1	Open the target project in Concept.
2	Call **File** →**Export** →**Program: section(s)**.
3	In the window **Open file** select the source project, e.g. C:\SOURCE_DIR\SOURCE.PRJ
4	Select the sections to be exported from the source project.
5	In **Save section export under** , specify the name of the export file (*.SEC), e.g. C:\TARGET_DIR\TARGET.SEC **Reaction:** The sections are exported and saved in the *.SEC file, e.g. in TARGET.SEC. The question **Import section into project now ?** follows
6	If the question as to whether the sections should be imported is answered with **OK** , the import will be performed now. Answer **Cancel**, if you want to start the mport later, see also procedure Resuming following canceled import *(see page 708)*.
7	Answer the question as to whether the project should be saved first with **OK**. **Note:** The query **Save project first ?** should be answered with **OK** , because, in the event of an import error, the current project is closed and all changes since the last save are rejected.
8	If it is required or necessary, it is possible in the table **Replace** , to make replacements for item names of variables, sections etc., as well as to define address offsets for located variables and direct addresses (refer to *Specific Changes, page 700*).

Step	Action
9	Select **OK** to continue (the whole import process is canceled if **Cancel** is selected). **Reaction:** Sections, used DFBs, used derived data types and the declarations for used variables, including comments, are imported into the target project. The import is canceled and the current project closed, if ● the sections to be imported contain DFBs that are not available in the target project. ● the sections to be imported contain DFBs whose versions differ from already available DFBs. (The imported DFB version can be accepted or rejected.) ● other errors arise during import. Errors are displayed in the messages window and have to be acknowledged.
10	If the import had been canceled, eliminate the cause of the cancelation and carry out the Resuming after import cancelation *(see page 708)*procedure.

DFB export and DFB import

To section export a source DFB and to then section import into a target DFB, carry out the following procedure:

Step	Action
1	Open the target DFB in Concept DFB.
2	Call **File** →**Export** →**Program: section(s)**.
3	In the window **Open file** select the source DFB, e.g. C:\SOURCE_DIR\SOURCE.DFB
4	Select the sections to export from the source DFB.
5	In **Save section export under** specify the name of the export file (*.SEC), e.g. C:\TARGET_DIR\DFB\TARGET.SEC **Reaction:** The sections are exported and saved in the *.SEC file, e.g. in TARGET.SEC. The question **Import section into project now?** is now displayed.
6	If this question is answered with **OK**, the import is performed now. If the answer given is **Cancel**, the import is started later, refer to Resuming after import break *(see page 708)*procedure.
7	Respond to the question as to whether the project should first be saved with **OK**. **Note:** The query **Save project first ?** should be answered with **OK** , because, in the event of an import error, the current project is closed and all changes since the last save are rejected.
8	If required or necessary, it is possible in the table **Replace**, to replace item names of variables, sections etc., as well as to define address offsets for located variables and direct addresses (refer to *Specific Changes, page 700*).

Step	Action
9	Select **OK** to continue (the whole import process is canceled if **Cancel** is selected). **Reaction:** Sections, used DFBs, used derived data types and the declarations for used variables, outputs and inputs are imported into the target project. The import is canceled and the current DFB closed, if ● the sections to be imported contain DFBs that are not available in the target DFB. ● the sections to be imported contain DFBs whose versions differ from already available DFBs. (The imported DFB version can be transferred or rejected.) ● other errors arise during import. Errors are displayed in the messages window and have to be acknowledged.
10	If the import had been canceled, eliminate the cause of the cancel and carry out the Resuming after import cancelation *(see page 708)*procedure.

Section export and DFB import

To section export a source project and to then section import into a target DFB, carry out the following procedure:

Step	Action
1	In Concept, delete all declarations of direct addresses and located variables in the sections to be exported. (They are not authorized in a DFB.)
2	Open the source project in Concept.
3	Call **File** →**Export** →**Program: section(s)**.
4	In the window **Open file** select the source project, e.g. C:\SOURCE_DIR\SOURCE.DFB
5	Select the sections to be exported from the source project.
6	In **Save section export under** specify the name of the export file (*.SEC), e.g. C:\TARGET_DIR\TARGET.SEC **Reaction:** The sections are exported and saved in the file *.SEC, e.g. in TARGET.SEC. The question **Import section into project now?** is now displayed.
7	Answer the question as to whether the sections should be imported with **Cancel**.
8	Close Concept.
9	Open Concept DFB and the target DFB.
10	Execute the menu command **File** →**Import** →**Program: section(s)**.
11	Select the export file (e.g. TARGET.SEC)
12	Respond to the question as to whether the project should firstly be saved with **OK**. **Note:** The question **Save project first ?** should be answered with **OK**, because in the event of an import error, the current project is closed and all changes made since the last save are rejected.

Step	Action
13	If required or necessary, in the table **Replace**, it is possible to replace item names of variables, sections etc., as well as to define address offsets for located variables and direct addresses (refer to *Specific Changes, page 700*).
14	Select **OK** to continue (the whole import process is canceled if **Cancel** is selected). **Reaction:** Sections, DFBs used, derived data types used and the declarations of used variables, inputs and outputs are imported into the target DFB. The import is canceled and the current DFB closed, if ● the sections to be imported contain DFBs that are not available in the target DFB. ● the sections to be imported contain DFBs whose versions differ from those of DFBs already available. (The imported DFB version can be transferred or rejected). ● other errors arise during import. Errors are displayed in the messages window and have to be acknowledged.
15	If the import had been canceled, eliminate the cause of the cancel and carry out the Resuming after import cancelation *(see page 708)*procedure.

DFB export and section import

To section export a source DFB and to then section import into a target project, carry out the following procedure:

Step	Action
1	Delete the input/output declarations in the DFB to be exported before exporting into Concept DFB, as these are not authorized in Concept projects.)
2	Open the source DFB in Concept DFB.
3	Call **File →Export →Program: section(s)**.
4	In the window **Open file** select the source DFB, e.g. C:\SOURCE_DIR\DFB\SOURCE.DFB
5	Select the sections to export from the source DFB.
6	In **Save section export under** specify the name of the export file (*.SEC), e.g. C:\TARGET_DIR\TARGET.SEC **Reaction:** The sections are exported and saved in the file *.SEC, e.g. in TARGET.SEC. The question **Import section into project now?** is now displayed.
7	Respond to the question as to whether the sections should be imported with **Cancel**.
8	Close Concept DFB.
9	Open Concept and the target project.
10	Execute the menu command **File →Import →Program: section(s)**.
11	Select the export file (e.g. TARGET.SEC).

Step	Action
12	Respond to the question as to whether the project should firstly be saved with **OK**. **Note:** The question **Save project first ?** should be answered with **OK**, because in the event of an import error, the current project is closed and all changes made since the last save are rejected.
13	If required or necessary, it is possible in the table **Replace**, to replace item names of variables, sections etc., as well as to define address offsets for located variables and direct addresses (refer to *Specific Changes, page 700*).
14	Select **OK** to continue (the entire import process is canceled if **Cancel** is selected). **Reaction:** Sections, DFBs used, derived data types used and the declarations of variables used, incl. comments, are imported into the target project. The import is canceled and the current project closed, if ● the sections to be imported contain DFBs that are not available in the target project. ● the sections to be imported contain DFBs whose versions differ from those of DFBs already available. (The imported DFB version can be transferred or rejected.) ● other errors arise during import. Errors are displayed in the messages window and have to be acknowledged.
15	If the import had been canceled, eliminate the cause of the cancel and carry out the Resuming after import cancelation *(see page 708)*procedure.

Resuming after import cancelation

To resume after an import cancelation, carry out the following procedure:

Step	Action
1	Open the target project/target DFB again.
2	Execute the menu command **File** →**Import** →**Program: section(s)**.
3	Select the export file (e.g. TARGET.SEC).
4	Answer the question **Back up project?:** with **Yes**. **Note:** The question **Back up project?** should be answered with **Yes**, because in the event of an import error, the current project is closed and all changes made since the last save are rejected.
5	If required or necessary, it is possible in the table **Replace**, to replace item names of variables, sections etc., as well as to define address offsets for located variables and direct addresses (refer to *Specific Changes, page 700*).
6	Select **OK** to continue (the whole import process is canceled if **Cancel** is selected). **Reaction:** Sections, DFBs used, derived data types used and the declarations of variables used, incl. comments, are imported into the target project. The import is canceled and the current project closed, if ● the sections to be imported contain DFBs that are not available in the target project/target DFB. ● the sections to be imported contain DFBs whose versions differ from those of DFBs already available. (The imported DFB version can be transferred or rejected.) ● other errors arise during import. Errors are displayed in the messages window and have to be acknowledged.

Importing IL and ST Programs to FBD, SFC, IL or ST Sections (with Conversion)

Introduction

Using **File →Import →Program: IEC text** ASCII files with IL or ST programs can be imported to FBD, SFC, IL or ST sections. ST and IL are able to have SFC elements (when imported into SFC section). Both text languages must adhere to the grammar of IEC text languages, shown in IEC 1131-3 and in the process tables 52 56 of IEC 1131-3.

Import units

The minimum import unit is a program organization unit (POU) to IEC (PROGRAM END_PROGRAM; FUNCTION_BLOCK ... END_FUNCTION_BLOCK).

The ASCII file can contain several POUs in Concept. From one POU, one or more sections bearing the same name as the POU arise, which is provided with a current number. A new section will be begun if too little graphic space is available to store the logic. FUNCTION_BLOCK ... END_FUNCTION_BLOCK-POUs are imported as DFBs.

In Concept DFB, the ASCII file can only contain a single POU. From this POU (FUNCTION_BLOCK END_FUNCTION_BLOCK) one section arises.

Inserting POUs:

Type of POU	Import into open project	Import into open DFB
PROGRAM ... END_PROGRAM	as a section into the current project.	not possible
FUNCTION_BLOCK ...END_FUNCTION_BLOCK	as project DFB. More than one POU can be imported at the same time.	as a section into the current DFB. Only one POU can be imported.
FUNCTION ... END_FUNCTION	is changed as DFB. The function name becomes the DFB output	is changed as DFB. The function name becomes the DFB output.

Behavior in the Event of Error

In this case, sections are only stored if the ST/IL text is syntactically perfect. POUs that cannot be reproduced are ignored completely, and an error message is displayed in the message window.

NOTE: If the file to be imported contains more than 200 declarations (declarations of variables and FFBs, a program error is caused. If this is the case, the declarations should be divided amongst several VAR...END_VAR blocks.

Variables

The variables declared in POUs appear after the import in the Variable Editor (exceptions: SFCSTEP_STATE and SECT_CTRL type variables).

EFBs with extended parameter set

EFBs with extended parameter set (PRE_DIA, GRP_DIA, LOOKUP_TABLE, ..) are only supported up to the predefined number of inputs/outputs.

"Bracket function" with extended number of inputs

If calls from a "bracket function" with extended number of inputs, such as MUX_INT() are imported then all instances of this function work with the maximum number of inputs that occur.

Changing from IL/ST to FBD

The following restrictions occur when changing to FBD:

- The following restrictions occur when changing to FBD:
- Block items can only be called once
- only assignments and block calls
  but none:
  - RET (table 52, property 20)
  - ELSIF (table 56, property 4)
  - ELSIF (table 56, property 4)
  - CASE (table 56, property 5)
  - FOR (table 56, property 6)
  - REPEAT (table 56, property 8)
  - EXIT (table 56, property 9)
- IF not nesting (IEC 1131-3 table 56, property 4)

Changing from IL/ST to SFC

The following limitations should be noted when making a SFC import from a text file:

- Only variables are permitted as actions. Direct addresses cannot be imported.
- Only literals are allowed as time variables for identifiers.
- Transition section names are changed to standard names.
- Step monitoring times and step delay times are lost when importing.

The following additional restrictions occur when changing to SFC (table = IEC 1131-3-table):

- Transitions conditions are stored in special FBD sections (TC_secname) (table 41, property 7a ,7c, 7d). The textual import of transition conditions is not possible.
- Actions are converted into FBD sections and not linked to steps.
- no identifier SD and SL (table 45, property 8, 10), they are imported as MOVE.

- Structure components and directly addressed variables are allowed as SFC actions. This can be seen as an extension of the IEC 1131-3 standard. ST and IL exports support neither.
- Using step variables ’step.X’ , ’step.T’ cannot be imported or exported and must be generated again.

Changing from IL/ST to ST or IL

The following restrictions apply when changing to ST or IL, that were not created in Concept.

- FB, DFB and direct address declarations occur at the start of the section (VAREND_VAR)
- the source formatting (indents, comments etc) applied only to the "logic part" of the sections, i.e. no comments for declarations (VAREND_VAR), for example
- Function Block counters must be made consistent, e.g. CTU must be changed to CTU_INT
- **no** Keywords
  - TYPE_...END_TYP
  - VAR_INPUT...END_VAR
  - VAR_OUTPUT...END_VAR
  - VAR_IN_OUT...END_VAR
  - VAR_EXTERNAL...END_VAR
  - FUNCTION...END_FUNCTION
  - FUNCTION_BLOCK...END_FUNCTIONBLOCK
  - PROGRAM...END_PROGRAM
  - STEP...END_STEP
  - TRANSITION...END_TRANSITION
  - ACTION...END_ACTION
- **no** RETURN instruction (ST Editor)
- **no** RET instruction (IL Editor)

Changing to Variable Declarations

When importing variable declarations the following restrictions occur:

- No comments are imported.
- VAR_CONSTANT is imported as located variable.
  ```
  (VAR_CONSTANT
  i : INT := 10;
  END_VAR
  ```
  becomes located variable "I" with the initial value of "10")
- VAR_INPUT and VAR_OUTPUT definitions are imported into the programs as located variables (VAR).
- VAR_INPUT and VAR_OUTPUT definitions are imported into DFBs as input/output variables (VAR_INPUT, VAR_OUTPUT).

Importing (insert file) IL and ST programs into IL or ST sections

At a Glance

Using **Edit →Insert text file...** it is possible to import ASCII files with IL or ST programs to IL or ST sections.

This import function is a pure text import function, which can also be performed via the clipboard (cut/copy/paste). Data conversion does not take place. For this reason, the necessary variable declarations for example (also if these are contained in the ASCII file) are not automatically integrated into the Variable Editor. The necessary variable declarations must be imported explicitly via **File →Import** from a "variables file", or be newly created. If variable declarations are contained in the section, they must be deleted, since they generate errors in the code generation of the section. Apart from this, all information for the POU must be deleted from the program (e.g. from the export of a graphic section using **File →Export →Program: IEC text**).

Restrictions

When importing IL and ST programs the following restrictions occur:

- no keywords
  - TYPE_...END_TYP
  - VAR_INPUT...END_VAR
  - VAR_OUTPUT...END_VAR
  - VAR_IN_OUT...END_VAR
  - VAR_EXTERNAL...END_VAR
  - FUNCTION...END_FUNCTION
  - FUNCTION_BLOCK...END_FUNCTIONBLOCK
  - PROGRAM...END_PROGRAM
  - STEP...END_STEP
  - TRANSITION...END_TRANSITION
  - ACTION...END_ACTION
- VAR...END_VAR
  - only for Function Block declarations and DFBs
  - only at the start of the section for all Function Blocks and DFBs in the section
  - not for variable declarations
  - apart from this, for making direct addresses consistent: VAR %Q10:INT; END_VAR
- **no** RETURN instruction (ST Editor)
- **no** RET instruction (IL Editor)

Procedure for "Copying" an IL section from an existing project into a new project.

Procedure

To "copy" an IL section from an existing project into an IL section of a new project, perform the following steps:

Step	Action
1	Open the IL section to be exported.
2	Using **Edit** →**Save as text file...** from the menu.
3	Select a directory for the export file and give it a name. Confirm with **OK**. **Reaction:** The IL section contents are copied into a new ASCII file.
4	Execute the menu command **File** →**Export** →**Variables: Text delimited**.
5	Select the filter settings **Export variables** and **Export constants**. Select comma as the text delimiter. Confirm with **OK**.
6	Select a directory for the export file and give it a name. Confirm with **OK**. **Reaction:** The variable declarations of your project are exported to an ASCII file.
7	Using **File** →**New project** generate a new project.
8	Using **Project** →**Configuration** open the configurator.
9	Using **Configure** →**PLC type** select a PLC. Confirm with **OK**.
10	Using **File** →**New section** generate an IL section.
11	Using **Edit** →**Insert text file...** import the IL file.
12	Using **File** →**Import** →**Variables: Text delimited**(**Warning:** Text delimiter must again be comma), import the variables file into the project's Variable Editor.
13	Check the import process using **Project** →**Analyze section**. **Reaction:** The import process is now completed and the new project can be edited in the normal way (Create further sections, complete the configuration etc.)

Procedure for converting FBD sections from an existing project into IL sections of a new project.

Procedure

The process of converting FBD sections from an existing project into IL sections in a new project consists of three main steps:

Step	Action
1	Exporting FBD section *(see page 714)*.
2	Importing FBD section into an IL section *(see page 714)*.
3	Correcting the syntax *(see page 715)*.

Exporting FBD section.

The procedure for exporting the FBD section is as follows:

Step	Action
1	Open the existing project.
2	Export the desired FBD section using **File** →**Export...** →**Program: IEC text**.
3	Select a directory for the export file and give it a name. Confirm with **OK**. **Reaction:** The FBD section is exported into an ASCII file.
4	Execute the menu command **File** →**Export** →**Variables: Text delimited**.
5	Select the filter settings **Export variables** and **Export constants**. Select comma as the text delimiter. Confirm with **OK**.
6	Select a directory for the export file and give it a name. Confirm with **OK**. **Reaction:** The variable declarations are exported to an ASCII file.

Importing FBD section into an IL section

The procedure for importing the FBD section into an IL section is as follows:

Step	Action
1	Using **File** →**New project** generate a new project.
2	Using **Project** →**Configuration** open the configurator.
3	Using **Configure** →**PLC type** select a PLC. Confirm with **OK**.
4	Using **File** →**New section** generate an IL section.
5	Using **Edit** →**Insert text file...** import the IL file.
6	Using **File** →**Import** →**Variables: Text delimited**(**Warning:** Text delimiter must again be comma), import the variables file into the project's Variable Editor. **Reaction:** The FBD section (in IL format) and the variable declarations were imported.

Correcting the syntax

The procedure for correcting the syntax is as follows:

Step	Action
1	Delete the line PROGRAM. (It contains the name of the old project.)
2	Delete any lines between VAR and END_VAR which do not contain Function Block or DFB declarations (e.g. variable declarations).
3	Delete all lines from INITIAL_STEP to END_STEP. (They contain the sections processing sequence of the old project.)
4	Change the ACTION lines to comment lines, e.g. (* ACTION xxx *). (They contain the names of the FBD sections.)
5	Delete the END_ACTION line.
6	Delete the END_PROGRAM line.
7	Verify the import process using **Project** →**Analyze section** and correct any errors. **Reaction:** The import process is now completed and the new project can be edited in the normal way (Create further sections, complete the configuration etc.)

21.5 Variables import

Overview

This section describes the importing of variables.

What's in this Section?

This section contains the following topics:

Topic	Page
Importing Variables in "Text Delimited" Format	717
Importing structured variables	720
Importing variables in Factory Link format	724
Multiple Address Assignment after Variable Import	725

Importing Variables in "Text Delimited" Format

Introduction

Using **File** →**Import** →**Variables: Text Delimited**, the variable declarations can be imported from an ASCII file into the variable editor in text delimited format.

Importing Initial Values

Initial values of variables in derived data types cannot be imported with this import format. If you wish to import initial values of variables in derived data types, select the IEC text import export/import format.

General Format Description

An ASCII file in "text delimited" format must conform to the following conditions:

- The character set used conforms to ANSI (Windows).
- The parameters of a variable are executed within one line.
- The individual parameters are separated from one another by a user-defined character.
- Leading and following spaces are allowed in any field (Exception: if a space has been used as a separator), the import function deletes the latter (with the exception of the comment field).
- The selected separator must not be contained in the individual parameters.
- Concept is not case-sensitive, in accordance with IEC name conventions. This should be adhered to for variable names.
- Overlapping between pre-existing addresses and addresses to be imported can be prevented in the following way: in the **Options** →**Preferences** →**Analysis...** →**Analysis Preferences** dialog, activate the **Treat Overlap of Addresses as an Error** option.

Order of Parameters within a Line

Order of Parameters within a Line:

- Variable flag
- Variable name (symbolic name)
- Data type
- Hardware address
- Initial value
- Comment

Meaning of Variable Flags

Possible values for the variable flags are:

- 0 or N= the symbolic name refers to a non-exportable variable
- 1 or E= the symbolic name refers to an exportable variable
- 2 or C= the symbolic name refers to a constant
- 3 or I = the symbolic name refers to an Input *(see page 487)* (Concept DFB only)
- 4 or O = the symbolic name refers to an Output *(see page 487)* (Concept DFB only)
- 5 or M = the symbolic name refers to a VARINOUT variable *(see page 489)* (Concept DFB only)
- S = Structured variable, see *Importing structured variables, page 720*.

Only variables with the 0/N or 1/E variable flag value are imported as located variables. All others are imported as unlocated variables.

If the variable flag is set at 2/C, the hardware address is ignored.

The values 3/I and 4/O are only permitted in Concept DFB. In this case, the values of the address fields are used for the position of the corresponding inputs and outputs. The variable flag value 1/E is imported into Concept DFB as variable flag value 0/N.

Structure of the Hardware Address Field

Structure of the Hardware Address Field (Example: %4:100):

- Characters for direct addresses "%" (may be missing)
- Address type
  - 0 = output, discrete
  - 1 = input
  - 3 = input word
  - 4 = output word, discrete word
- Separator ":" or ".".
  If no separator is used, the address must be 6 characters long.
- Address

Examples of an Address Description

Output register 123 :

- %400123 or
- %4.123 or
- %4:123 or
- 400123 or
- 4.123 or
- 4:123

IEC Address Conventions

The IEC address conventions can also be used (e.g. %QX100 corresponds to 000100):

Address Type	Concept Designation	IEC Designation
Output, discrete	0x	%QX,%Q
Input	1x	%IX,%I
Input register	2x	%IW
Output register, discrete register	3x	%QW

Empty Fields

Empty fields are represented by two consecutive separators.

The following fields are allowed to be empty:

- Hardware address
- Initial value
- Comment

Missing Fields

The following fields are allowed to be missing:

- Comment
- Comment and initial value
- Comment and initial value and hardware address

Importing structured variables

At a Glance

The basic structure of the file corresponds to that of the variables in text delimited *(see page 717)* format.

Additional usage designations

In addition, the following points should be taken into account:

- Multiple rows are necessary to describe a variable.
- Each of these rows must correspond to the format of variables in delimited text format.
- A structured variable with initial values is described by an introducing row with the following structure:
  a. Variable flag
  b. Variable name (symbolic name)
  c. Name of derived data type
  d. Hardware address
  e. Empty field
  f. Comment
- This introductory line is followed by at least one component description. This component description results from the description of the element components (element data type) in the form of a row with the following structure (a component does not have to be described if its initial value is the same as the standard value). The sequence in which the individual components are executed is insignificant.
  a. Character "S" (S stands for structured)
  b. Path of components (the variable name does not have to be included)
  c. Field for IEC data type (this field can remain empty)
  d. Empty field
  e. Initial value
  f. Empty field

Component description error trapping

Component description error trapping

- If a variable component is described more than once, the last description is used.
- If the specified component is not contained in the currently described variable, the component description is ignored and a warning is given.
- If the field for the components path is empty, the component description is ignored and a warning is given.
- If the field for the IEC data type is not empty, the specified data type is checked. If the specified data type and the data type of the component are not the same, the component description is ignored and a warning is given.

- Entries in the address field are ignored.
- Entries in the address field are ignored.

Example: Structured variables in "Text delimited" format

Structured data type definition ESI_IN:

```
ESI_In:   (* ESI - input data *)
  STRUCT
    in:          ESI_InOut;     (* ESI input data *)
    esi:         ESI_Status;
    dummy:       BYTE;          (* supplement to modulo 16 *)
    slot:        Exp_Status;
  END_STRUCT;


ESI_InOut:  (* ESI input / output data structure *)
  STRUCT
    tstat:       BYTE;     (* transfer status, handshake *)
    blocks:      BYTE;     (* number of used blocks *)
    res:         BYTE;     (* reserved *)
    block:       ESI_BlockArr14;     (* data block *)
  END_STRUCT;


ESI_BlockArr14: ARRAY[1..14] OF ESI_Block;


ESI_Block:  (* datas of ESI *)
   STRUCT
    func:        BYTE;     (* function *)
    mux:         WORD;     (* distribution *)
    attr:        BYTE;     (* attribute *)
    cause:       BYTE;     (* reason *)
    station:     WORD;     (* station number *)
    object:      WORD;     (* objekt number *)
    data:        ByteArr9; (* data bytes *)
   END_STRUCT;


ByteArr9:        ARRAY [1..9] OF BYTE;    (* 9 bytes *)
```

```
ESI_Status:  (* Status of ESI *)
   STRUCT
    wdog:        BYTE;     (* expert watchdog-counter *)
    stat1:       BYTE;     (* error status 1 *)
    stat2:       BYTE;     (* error status 2 *)
    stat3:       BYTE;     (* error status 3 *)
    slot:        WORD;     (* slot number *)
    user:        WORD;     (* virtual slot number *)
    esitime:     DPM_Time; (* time stamp *)
  END_STRUCT;


DPM_Time:  (* time stamp *)
  STRUCT
    sync:        BOOL;     (* sync clock *)
    ms:          WORD;     (* milli-seconds *)
    min:         BYTE;     (* minutes *)
    hour:        BYTE;     (* hours; (hour AND 16#80) *)
                           (* = day light saving time *)
    day:         BYTE;     (* days of week *)
    mon:         BYTE;     (* month *)
    year:        BYTE;     (* year *)
  END_STRUCT;


STRUCT
  Exp_Status:  (* error status of transfer *)
    ErrFlag1:    BOOL;     (* TRUE: epxert not pluged *)
    ErrFlag2:    BOOL;     (* TRUE: Bit 7 of DPM *)
                           (* Identcode is set; *)
                           (* logical DMP-access-error *)
    UserStatus:  WORD;     (* status of expert *)
    ErrNo:       WORD;     (* errornumber *)
END_STRUCT;
```

Representation of variables "demo" of ESP_IN data type in delimited text format:

```
1;demo;ESI_In;400002;;structured data type
S;in.tstat;BYTE;;16#0F;
S;in.blocks;BYTE;;16#0F;
S;in.res;BYTE;;16#0F;
S;in.block[1].func;BYTE;;16#0F;
S;in.block[1].mux;WORD;;16#000F;
S;in.block[1].attr;BYTE;;16#0F;
S;in.block[1].cause;BYTE;;16#0F;
S;in.block[1].station;WORD;;16#000F;
S;in.block[1].object;WORD;;16#000F;
S;in.block[1].data[1];BYTE;;16#0F;
S;in.block[1].data[5];BYTE;;16#0F;
S;in.block[3].func;BYTE;;16#0F;
S;in.block[3].mux;WORD;;16#000F;
S;in.block[3].func;BYTE;;16#0F;
S;in.block[3].cause;BYTE;;16#0F
S;in.block[3].station;WORD;;16#000F
S;in.block[3].object;WORD;;16#000F
S;in.block[3].data[1];BYTE;;16#0F
S;in.block[3].data[2];BYTE;;16#0F
S;esi.wdog;BYTE;;16#0F
S;esi.stat1;BYTE;;16#0F
S;esi.stat2;BYTE;;16#0F
S;esi.stat3;BYTE;;16#0F
S;esi.slot;WORD;;16#000F
S;esi.user;WORD;;16#000F
S;esi.esitime.sync;BOOL;;TRUE
S;esi.esitime.ms;WORD;;16#000F
S;esi.esitime.min;BYTE;;16#0F
S;esi.esitime.hour;BYTE;;16#0F
S;esi.esitime.day;BYTE;;16#0F
S;esi.esitime.mon;BYTE;;16#0F;
S;esi.esitime.year;BYTE;;16#0F;
S;dummy;BYTE;;16#0F;
S;slot.ErrFlag1;BOOL;;FALSE;
S;slot.ErrFlag2;BOOL;;FALSE;
S;slot.UserStatus;WORD;;16#000F;
S;slot.ErrNo;WORD;;16#000F;
```

Importing variables in Factory Link format

Description

Using **File →Import →Variables: Factory Link** variable declarations in Factory Link format can be imported. In addition, carry out a Factory Link export and specify the Factory Link version when importing into Concept.

If your Factory Link version of Concept is not supported, please call our hotline.

NOTE: Factory Link is case-sensitive with variable names. Concept does not differentiate in accordance with IEC naming conventions. This should be adhered to during import

Multiple Address Assignment after Variable Import

Description

Via a Variables Import the multiple assignment of a single address by different variable names is possible. This situation occurs if prior to the import, a variable that has already been used in Concept is renamed in the list to be imported. In order to not have to also manually perform this renaming in Concept, after the import you can open the dialog box **Multiple Address Assignment** and perform the renaming or replacement of the variable names automatically in the entire project (e.g. in Variable Editor, Sections).

NOTE: In large projects and a corresponding number of multiple assignments, the updating of the variable names can take some time.

21.6 Import/Export of PLC Configuration

Overview

This Section describes the import and export of the PLC configuration with Concept or Concept Converter.

What's in this Section?

This section contains the following topics:

Topic	Page
Import/Export of PLC Configuration using Concept	727
Import/Export of PLC Configuration using Concept Converter	728

Import/Export of PLC Configuration using Concept

Introduction

Using the Import/Export function a PLC configuration can be exported out of a current (open) project and subsequently re-imported.

Config. Export and Config. Import

To export and subsequently import SPS configurations, proceed as follows:

Step	Action
1	To export the PLC configuration from the current project, start Concept, open the desired project and select **File** →**Export** →**Configuration**.
2	In the **Folder** field, select the target directory for the PLC configuration to be exported.
3	In the **File name** field, enter a name for the Export file (NAME.CCF) and press **OK**. **Response:** The PLC configuration is stored in the selected directory as an ASCII file.
4	To import a PLC configuration into a project, open the desired project.
5	In Concept select the **File** →**Import** →**Configuration**menu command.
6	From the **File type** list select the **Concept Configuration entry. (*.CCF)**.
7	In the **Folder** field, select the desired directory.
8	From the **File name** list select the PLC configuration to be imported (NAME.CCF) and click on **OK**.
9	**Warning:** The current PLC configuration of the chosen project will be overwritten. Answer the question with **OK**. **Response:** The PLC configuration is imported.

Import/Export of PLC Configuration using Concept Converter

Introduction

The Concept Converter's import/export function enables you to export the configuration from one project (Project A) and import it into another project (Project B).

Config Export and Config Import

In order to export and then import a PLC configuration, carry out the following steps:

Step	Action
1	To export the PLC Configuration from project A, start the Concept Converter and select **File** →**Export** →**Configuration**.
2	From the **Folder** field, select the Project A system directory.
3	Select the PLC configuration to be exported (PROJECTNAME.C1) and click on **OK**. **Response:** The PLC configuration is filed in the system directory in the form of an ASCII file (PROJECTNAME.CON).
4	To import the PLC configuration into Project B, copy the exported file into the system directory of Project B.
5	In Concept Converter select the **File** →**Import** menu command.
6	From the **File Type** list box select the **Configuration (*.CON)** entry.
7	From the **Folder** field, select the Project B system directory.
8	From the **File Name** list box, select the PLC configuration (PROJEKTNAME.CON) to be imported and click on **OK**.
9	**Caution:** The current PLC configuration of the selected project will be overwritten. Answer the question with **OK**. **Response:** The PLC configuration will be imported.

Documentation and Archiving

22

Overview

This chapter describes the documentation, the archiving and deleting of projects, DFBs and macros.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
22.1	Documentation of projects, DFBs and macros	730
22.2	Managing projects, DFBs and macros	740

22.1 Documentation of projects, DFBs and macros

Overview

This section describes the documentation of projects, DFBs and macros.

What's in this Section?

This section contains the following topics:

Topic	Page
Documentation contents	731
Documentation Layout	732
Defining Page Breaks for Sections	735
Use of keywords	739

Documentation contents

At a Glance

The contents of the documentation can range in length from one on-screen page to the entire documentation of a project. The order in the first chapter is given as in the dialog box **File** →**Print** →**Documentation contents** and cannot be changed.

Project documentation

The following chapter can be printed for project documentation using the menu command **File** →**Print**:

- Project description
- Derived data types
- Using state RAM
- State RAM values
- Using the DFBs
- Using the EFBs
- PLC configuration
- I/O Map
- Execution sequence of the sections
- Project structure
- Messages
- ASCII messages only with Concept for Quantum
- Variable lists
- Use of variables
- Contents of sections
- Contents directory for the printed documentation

DFB/macro documentation

The following chapter can be printed for DFB/macro documentation using the menu command **File** →**Print** :

- DFB/macro description
- Derived data types
- Using the DFBs
- Using the EFBs
- Execution sequence of the sections
- Messages
- Variable lists
- Use of variables
- Contents of the sections
- Contents directory for the printed documentation

Documentation Layout

Print Format

The printout can be in either portrait or landscape mode. This is set up in the dialog box **File** →**Printer Setup** →**Select Printer**.

Page Numbering

The pages are numbered linearly. The starting page number can be selected by the user.

Page Size

The left margin is 12 characters wide. The area for text and graphics is approximately 132 characters wide, the height depends on the header and footer files. If the header and footer files are not activated or the keyword "%PAGENO" is not contained in them, the page number will be printed automatically in the bottom right corner of the page.

Page Breaks

If a graphics section does not fit on a printed page, the section will be divided - like a map - in the printout. In this case page references are printed in all four corners of the graphics area to show which page the graphics are continued on. The **View** → **Page Break** menu option displays the page break corresponding to the printer set in **File** →**Printer Setup** and to the enlargement factor in the editor window.

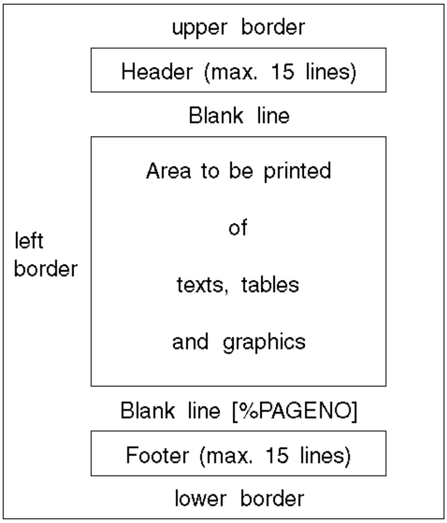
Also see the *Defining Page Breaks for Sections, page 735* description.

Size and Fonts

In text sections the font size in the printout cannot be altered. Emphasis of keywords is represented in the printout using bold and italic typefaces.

Standard Layout

Standard Layout:



Header

It is possible to give your documentation a header. The header is stored as an ASCII file and can be created using any ASCII editor. The maximum file size is 15 lines or approx. 2 Kbytes.

A sample file called "HEADER.TXT" is available in the Concept directory. This file can be modified as required. Keywords *(see page 739)* can also be used with it.

Footer

It is possible to give your documentation a footer. The footer is stored as an ASCII file and can be created using any ASCII editor. The maximum file size is 15 lines or approx. 2 Kbytes.

An sample file called "FOOTER.TXT" is available in the Concept directory. This file can be modified as required. Keywords *(see page 739)* can also be used with it.

Front Page

It is possible to give your documentation a front page. The front page is stored as an ASCII file and can be created using any ASCII editor. The size of the file is unlimited.

An sample file called "FRONTPG.TXT" is available in the Concept directory. This file can be modified as required. Keywords *(see page 739)* can also be used with it.

The printout of the front page also contains the header and footer if these are switched on.

Defining Page Breaks for Sections

Introduction

For printing graphics in FBD, LD and SFC sections, you can define the values for the page break and paper orientation of the graphics. The higher the value you select, the smaller the graphics will be displayed. But in return more space is available on a page.

Settings

You can set the values for the page break for portrait and landscape. When changing the paper format, the settings for the other format stay saved. Using the **Download standard values** command button, the standard values from the CONCEPT.INI file can be loaded.

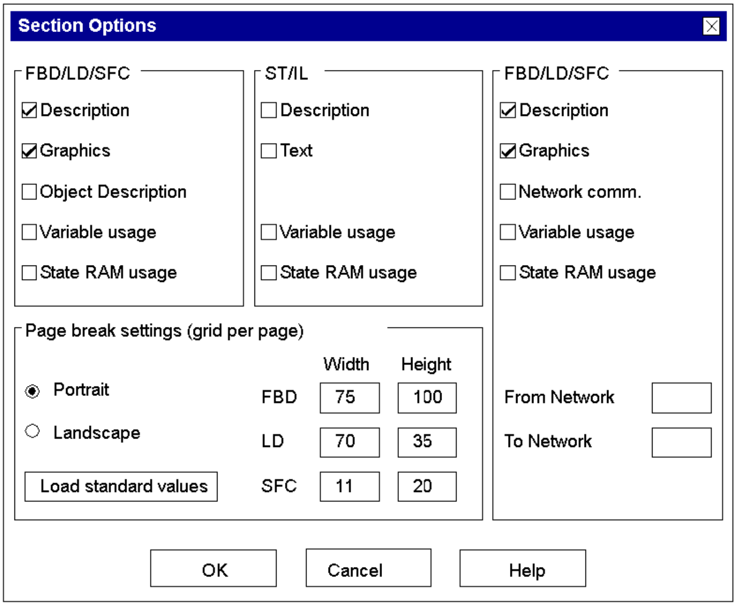
When defining values for the width and height of the paper, you should make sure that the different editors show different grid units.

The min. and max. values are:

Section	1 grid unit equals the value	Paper Width	Paper Height
FBD	10	30 - 300	30 - 230
LD	8	30 - 400	10 - 230
SFC	1	4 - 32	4 - 60

Example for FBD section

Dialog setting



Section Options

FBD/LD/SFC

☑ Description

☑ Graphics

□ Object Description

□ Variable usage

□ State RAM usage

ST/IL

□ Description

□ Text

□ Variable usage

□ State RAM usage

FBD/LD/SFC

☑ Description

☑ Graphics

□ Network comm.

□ Variable usage

□ State RAM usage

Page break settings (grid per page)

● Portrait

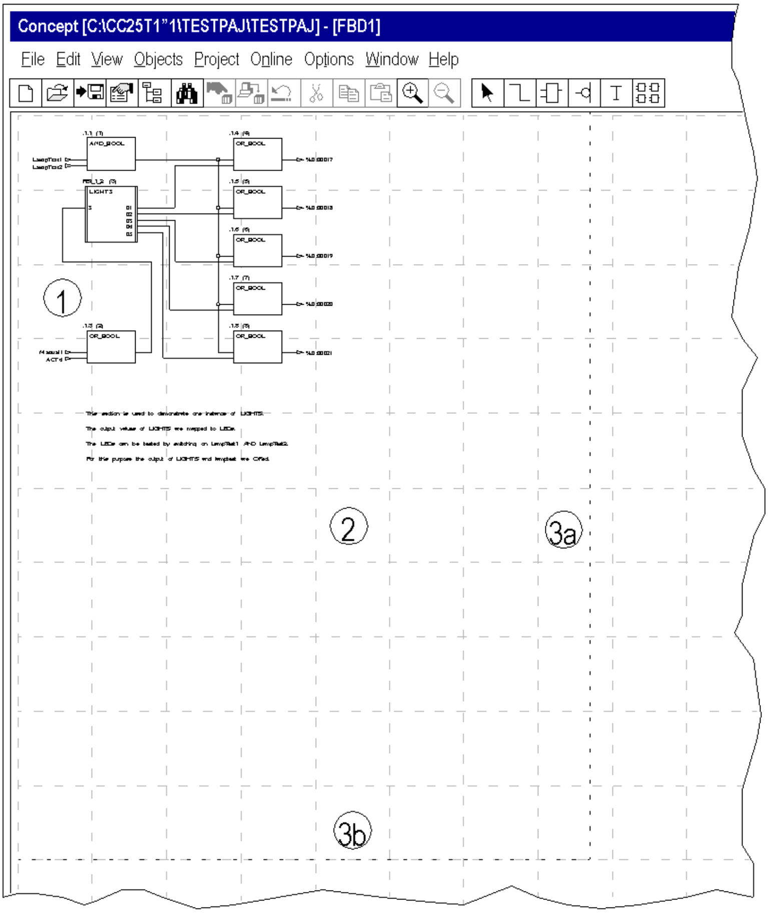
○ Landscape

	Width	Height
FBD	75	100
LD	70	35
SFC	11	20

From Network

To Network

Representation in the FBD editor window



Concept [C:\CC25T1"1\TESTPAJ\TESTPAJ] - [FBD1]

This section is used to demonstrate one instance of LIGHTS.

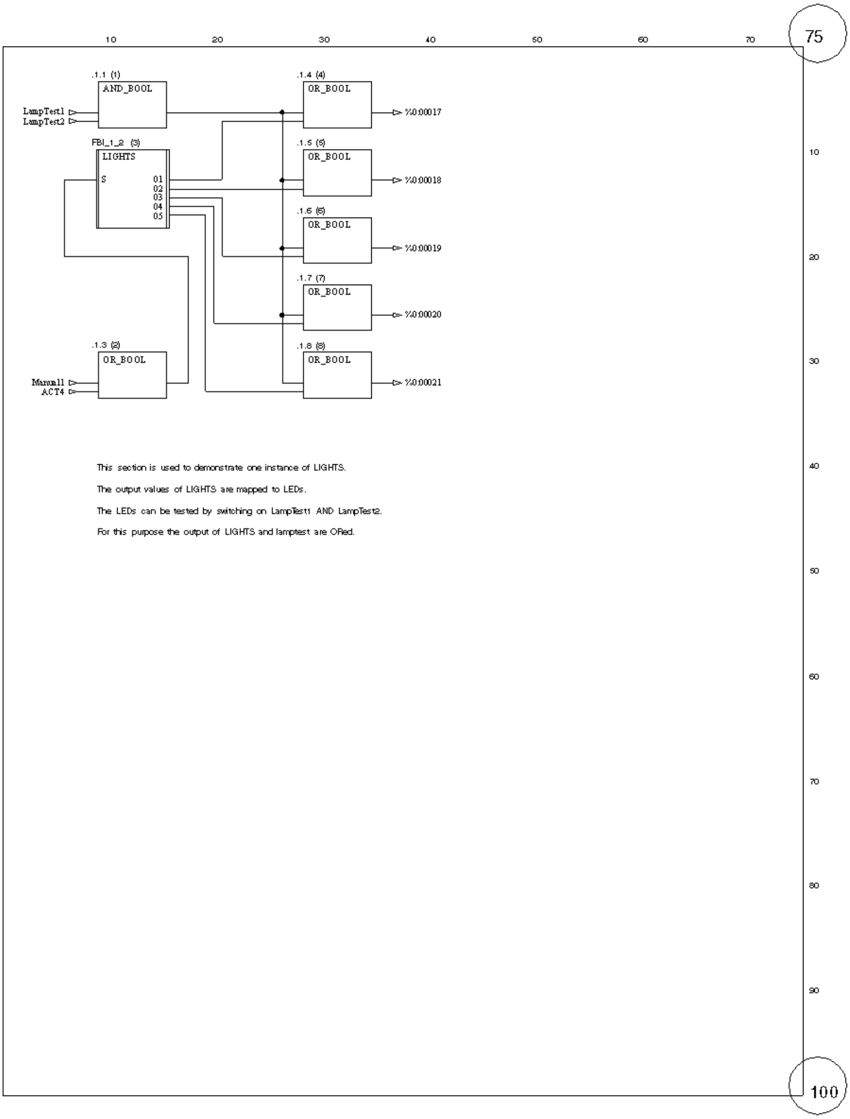
The output values of LIGHTS are mapped to LEDs.

The LEDs can be tested by switching on LampTest1 AND LampTest2.

For this purpose the output of LIGHTS and lamptest are ORed.

1 FBD section
2 Grid view (View -> Grid)
3a Page break, width: 75 (View -> Page break)
3b Page break, height: 100 (View -> Page break)

Print-out



This section is used to demonstrate one instance of LIGHTS.

The output values of LIGHTS are mapped to LEDs.

The LEDs can be tested by switching on LampTest1 AND LampTest2.

For this purpose the output of LIGHTS and lamptest are ORed.

Use of keywords

At a Glance

Keywords allow project or object specific information to be inserted into the header, footer and title page files.

Usable keywords

Table of usable keywords:

%PROJNAME	**Project name**
%SECTNAME	Section name
%VERSION	Program/DFB version
%CREDATE	Creation date
%MODDATE	Date of last project/DFB modification
%DATE_D	Current date (European format, DD.MM.YY)
%DATE_US	Current date (US format, DD.MM.YY)
%PAGENO	Current page numbers
%RECT(Column,Width,Height)	Draws a rectangle with its top left-hand corner in the current line
%HLINE(Column,Length)	Draws a horizontal line in the current line
%VLINE(Column,Length)	Draws a vertical line starting in the current line

NOTE: The total number of lines in the header, footer or title page file must agree with the number of lines needed to print rectangles and vertical lines.

Example: Header with keywords

Contents of the ASCII file:

```
%RECT (1,132,4)    %VLINE (24,4)    %VLINE (110,4)
     S A              Project comment           Name
   CONCEPT                                        %DATE_D
¶
```

NOTE: The symbol ¶ is not entered, it should only show that the file ends with a blank line.

Expression:

S A CONCEPT	Project comment	Name 01.04.99

22.2 Managing projects, DFBs and macros

Overview

This section describes the archiving and deletion of projects, DFBs and macros.

What's in this Section?

This section contains the following topics:

Topic	Page
Archiving projects, used DFBs, EFBs and data type files	741
Deleting projects, DFBs and macros	743

Archiving projects, used DFBs, EFBs and data type files

Introduction

When archiving projects, used DFBs, EFBs and data type files, all data of the project is collected and compressed. The *.PRZ file is created and put into the same directory as the project itself. The file can be decompressed at any time thereafter.

NOTE: When archiving DFBs, their help files (*.DOC, *.PDF, *.TXT), which are located in the Concept directory or the defined path (see CONCEPT.INI *(see page 1110)*) are not considered. However, if you want to archive these help files, you must copy these files into the local/global DFB-directory.

Archiving Projects

The procedure for archiving projects is as follows:

Step	Action
1	Start Concept. **Note:** No project may be open during the archiving procedure, otherwise the **Archive...**command cannot be selected.
2	To archive, select **File →Archive...**. **Response:** A window showing the Concept projects appears.
3	Select the project to be archived from the window and press **OK**. **Reaction 1:** A check for whether a compressed *.PRZ file has the same name is performed. If there is a file with the same name, you are requested to confirm whether the existing file should be replaced by the new file. **Reaction 2:** The project data is compressed and saved in the *.PRZ file and is then found in the same directory as the project.

Unpacking Archived Projects

The procedure for unpacking archiving projects is as follows:

Step	Action
1	Select **File →Open**. **Result:** A window showing all Concept projects appears.
2	Go to the list field **File Type** and select option **Archived Projects (*.prz)**. **Result:** The archived Concept projects are displayed.
3	Select the project that you want to open and press **OK**. **Reaction 1:** A check for whether a *.PRJ file has the same name is performed. If there is a file with the same name, you are requested to confirm whether the existing file should be replaced by the new file. **Reaction 2:** A check for whether DFBs, EFB libraries and data type files with the same name exist, is performed. If there is a file with the same name, you are requested to confirm whether the existing file should be replaced by the new file. **Reaction 3:** The **Archive content** dialog is opened.

Step	Action
4	Select **Decompress**. **Reaction 1:** The project data is decompressed and stored as a normal Concept project. The project is then found in the same directory as the archived file. **Reaction 2:** The project is automatically opened in Concept.
5	Establish a connection between the PC and the PLC with **Online** → **Connection**. **Result:** The PC and PLC have the same status as before the archive procedure.

Archiving/unpacking global DFBs

When archiving or unpacking the used global DFBs, the following sequence should be used:

Step	Action
1	The project directory is searched for an existing GLB directory.
2	The relevant settings are checked in the CONCEPT.INI file. For example: [Path]: GlobalDFBPath=x:\DFB [Upload]: PreserveGlobalDFBs=0 In this example, the DFB directory of the path defined is searched for global DFBs.
3	The DFB directory in x:\CONCEPT\DFB is searched.

The global DFBs from only one directory are used and/or are saved in only one directory. i.e. if step 1 is unsuccessful, then step 2 follows, step 3 is only performed if neither of the first two are successful.

Diagnostic Information

When downloading the project, diagnostic information is created and put into the corresponding directory. Then, the status between PC and PLC becomes EQUAL. When archiving the project, this diagnostic information is compressed with the other project data and stored in a file.

To use the diagnostic information after it is decompressed, make sure that the status between the PC and the PLC is EQUAL when archiving. Downloading is no longer required and diagnostics can be run immediately.

If the status is anything else between the PC and the PLC, e.g. NOT EQUAL, then this status will be displayed while decompressing and after the connection (**Online** →**Connect...**). Downloading is therefore required to put the system into operation. Downloading also creates new diagnostic information while the old information is lost however.

Deleting projects, DFBs and macros

Deleting projects, DFBs and macros

The procedure for deleting projects, DFBs and macros is as follows:

Step	Action
1	Delete the project/DFB/macro directory (including the subdirectory "dfb"). If only certain DFBs/macros need to be deleted from this directory, open the subdirectory and delete all files with the required DFB/macro name (name *).
2	Use global DFBs, and global macros in the project/DFB and if these also need to be deleted, they must be deleted separately. Open the subdirectory "dfb" of the Concept directory and delete all files carrying the name DFBs/macros (name *).

Simulating a PLC

23

Overview

This chapter describes how to simulate a PLC. By using a simulator the functions of a program may be tested without the actual required hardware.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
23.1	Simulating a PLC (16-bit simulator)	746
23.2	Simulating a PLC (32-bit simulator)	748

23.1 Simulating a PLC (16-bit simulator)

Simulating a Controller

Introduction

This section describes the 16-bit simulator Concept SIM.

Area of Application

Concept SIM may be used to simulate any PLC (Quantum, Compact, Momentum, Atrium) in order to test the user program "online" without hardware.

The simulator is only available for the IEC languages (FBD, SFC, LD, IL and ST).

The 16-bit simulator Concept SIM is used for testing programs containing Concept EFB generated 16-bit EFB.

NOTE: If your program does not contain 16-bit EFBs created with Concept EFB, you should use the 32-bit simulator (PLCSIM) to simulate a PLC.

Max. Number of Variables

When using the 16 bit simulator Concept SIM, a specific number of state RAM references (**Project** →**PLC configuration** →**Configuring** →**Memory Partitions**) may not be exceeded.

The table below shows the maximum number of these state RAM references:

Reference type	max. number
0x	60000
1x	5008
3x	4000
4x	24000

Concept vs. Concept SIM

Concept SIM and Concept may only be opened independently, i.e. when starting Concept SIM, Concept cannot be open. It is therefore advisable to decide before starting Concept, whether the simulator or the controller should perform the test. In each case, make sure that the simulator is turned on or off as required.

Activating Concept SIM

The procedure for activating Concept SIM is as follows:

Step	Action
1	Close Concept if it is open.
2	Open Concept-SIM by double-clicking on the Concept-SIM icon.
3	Click on the **File** main menu and activate the **Simulation on** menu command. **Response:** The simulator is on.
4	Exit Concept SIM via the **File** main menu using the **Exit** menu command.
5	Start Concept.
6	From **Online** →**Connect...** open the **Connect to PLC** dialog window.
7	For **Protocol type:** always select **Modbus Plus**, even if your real PLC will be coupled via a different bus at a later stage. **Response:** The simulator will now be displayed as a PLC in the node list of the Modbus Plus network.
8	Now create a link to the simulated PLC by double clicking on the list entry or via **OK**. **Response:** You may now test the behavior of your IEC user program.

Note

NOTE: Please note that the simulator remains active even after rebooting the PC. To build a link to a PLC the simulator must be explicitly terminated.

Disabling Concept SIM

The procedure for disabling Concept SIM is as follows:

Step	Action
1	Close Concept if it is open.
2	Open Concept-SIM by double-clicking on the Concept-SIM icon.
3	Click on the **File** main menu and select the **Simulation Off** menu command. **Response:** The simulator is on.
4	Exit Concept SIM via the **File** main menu using the **Exit** menu command.

23.2 Simulating a PLC (32-bit simulator)

Overview

This Section describes how to simulate a PLC with the 32-bit simulator Concept PLCSIM32.

What's in this Section?

This section contains the following topics:

Topic	Page
Concept-PLCSIM32	749
Simulating a PLC	751
Simulating a TCP/IP interface card in Windows 98	753
Simulating a TCP/IP interface card in Windows NT	754

Concept-PLCSIM32

Introduction

The Concept-PLCSIM32 program simulates any PLC unit (Quantum, Compact, Momentum, Atrium) and its signal states.

Area of Use

The simulator is presently only available for IEC languages (FBD, SFC, LD, IL and ST).

NOTE: Not supported:

- LL984 language
- Loadables, e.g. ULEX
- 6x-Register (extended memory)
- RIO
- DIO
- Backplane Expander

Note for Windows 98 and Windows NT

Since the simulator is connected to Concept via a TCP/IP link, you need a card in your computer to handle the TCP/IP interface (when using Windows 98 or Windows NT). If your computer is not equipped with such a card, it can be simulated. Follow the procedure described in Simulation of a TCP/IP interface card in Windows 98 *(see page 753)* or Simulation of a TCP/IP interface card in Windows NT *(see page 754)*.

When using Windows 2000, simulating a TCP/IP interface card is not necessary because all drivers needed for Concept PLCSIM32 are installed automatically.

Structure of the dialog box

The title bar shows the name of the application (PLC Sim32) and the address of your PC-interface card.

The first text box in the simulator window shows the status of the simulated PLC. This field is read-only. The displayed status is determined by Concept, as with a real PLC.

The status may be shown as the following:

- **DIM** (Dim Awareness)
  The simulator is in an undefined state.
- **STOPPED**
  The simulator (the simulated PLC) is stopped.
- **RUNNING**
  The simulator (the simulated PLC) is running.

The type of PLC to be simulated can be selected from the first list box.

The following registers are available:

- **State RAM**
  Provides an overview of the signal memory.
- **I/O Modules**
  Shows the configuration currently loaded or the signal memory of a selected group of components.
- **Connections**
  Displays connections between the simulator and programming device(s).

Simulating a PLC

Overview

A controller is simulated with the PLCSIM32 simulator using 4 main steps:

Step	Action
1	Program creation and controller configuration.
2	Activating the simulator.
3	Construction of the connection between Concept and simulator.
4	Downloading the program.

Program creation and controller configuration

The following steps describe how to create programs and configure the controller.

Step	Action
1	Create your program and your controller configuration in Concept.
2	Save your project with **File** →**Save**.

Activating the simulator

The following steps describe how to activate the simulator:

Step	Action
1	Run PLCSIM32 simulator in the Concept program group.
2	Select the controller type appropriate to your project in the simulator.

Construction of the connection

The following steps describe how to construct the connection between Concept and the simulator.

Step	Action
1	Using **Online** →**Connect...** open the **Connect to PLC** dialog in Concept.
2	Select the **IEC Simulator (32-Bit)** entry in the Protocol Type list box.
3	In the **Access** range, activate the **Change configuration** option button.
4	Confirm with **OK**. **Response:** A connection has been made between the programming unit and the simulator. A note then appears, saying that the configurations of the programming unit and the simulator are different.

Downloading the program

The following steps describe how to download the program:

Step	Action
1	Using **Online** →**Download** open the **Download Controller** dialog.
2	Confirm with **Download**. **Response:** Your program and your configuration are loaded into the simulator. You will be asked if you wish to start the controller.
3	Confirm with **Yes**. **Response:** You may now test the behavior of your IEC user program.

Simulating a TCP/IP interface card in Windows 98

Introduction

As the coupling between Concept and the simulator PLCSIM32 is made via a TCP/IP coupling, a TCP/IP interface card is needed in the PC. If your PC does not have one of these cards, it may be simulated.

⚠ CAUTION

Risk of PC problems

Do **NOT** complete this procedure if your PC already has a TCP/IP connection. The software installation of the TCP/IP connection would be destroyed by this procedure. Only carry out this procedure once, otherwise PC problems may arise.

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

Simulating a TCP/IP Interface Card

Carry out the following steps to simulate a TCP/IP interface card in Windows 98:

Step	Action
1	In Windows 98 invoke **Start** →**Settings** →**Control Panel**.
2	From **Software** open software settings.
3	From the **Windows Setup** register select the **Links** entry and click on the **Details...** command button.
4	Check the **DFU network** entry here and confirm with **OK**. (To do this, you may require the Windows system CD.) **Response:** The computer reboots. The DFU network and the TCP/IP protocol are available to the system after the reboot. (Concept can only connect to the simulator.)

Simulating a TCP/IP interface card in Windows NT

Introduction

As the coupling between Concept and the simulator PLCSIM32 is made via a TCP/IP coupling, a TCP/IP interface card is needed in the PC. If your PC does not have one of these cards, it may be simulated.

⚠ CAUTION
Risk of PC problems
Do **NOT** complete this procedure if your PC already has a TCP/IP connection. The software installation of the TCP/IP connection would be destroyed by this procedure. Only carry out this procedure once, otherwise PC problems may arise.
Failure to follow these instructions can result in injury or equipment damage.

Simulating a TCP/IP Interface Card

The main steps for simulating a TCP/IP interface card in Windows NT are as follows:

Step	Action
1	Setting the basic settings.
2	Installing a new modem.
3	Setting the workgroup.

Setting the Basic Settings

The procedure for setting the basic settings is as follows:

Step	Action
1	In Windows NT, invoke **Start** →**Settings** →**Control Panel** →**Network** and answer **Yes** to the question. **Response:** The **Network Setup Wizard** dialog is opened.
2	Deactivate the **Wired to the network** option.
3	Select the **Remote access to the network** option. **Response:** The network card installation dialog will be opened.
4	Click on **Next** (without installing a network card). **Response:** The dialog for selecting a network protocol will be opened.
5	Select the **TCP/IP-Protocol** option.
6	Deactivate all the other options and click on **Next**. **Response:** The dialog for selecting services will be opened.

Step	Action
7	Click on **Next** (without making any changes in the dialog).
8	Answer the question with **Next**. Response: The **Windows NT Setup** dialog is opened.

Installing a New Modem

The procedure for installing a new modem is as follows:

Step	Action
1	Insert the Windows NT CD and specify the path for the installation data files (for example `D:\i386`). Click on **Resume**. **Response:** The **TCP/IP Setup** dialog is opened.
2	Click on **No**. **Response:** The **Remote Access Setup** dialog is opened.
3	Click on **Yes**. **Response:** The **Install New Modem** dialog is opened.
4	Select the **Don't detect my modem; I will select it from a list.** option and press **Next**. **Response:** The dialog for selecting a modem is opened.
5	Select a standard modem (for example **Standard 28800 bps modem**) and press **Next**. **Response:** The dialog for selecting the connection is opened.
6	Select the **Selected ports** option and the **COM** interface. Click on **Next**. **Response:** The **Standard information** dialog is opened.
7	Select your country.
8	Enter the code for your town (your area code) and click on **Next**. **Response:** The **Install New Modem** dialog is opened.
9	Click on **Finish**. **Response:** The **Add Remote Access Setup device** dialog is opened.
10	Click on **OK**. **Response:** The **Remote Access Setup** dialog is opened.
11	Click on **Next**. **Response:** The **Network installation assistant** dialog is opened.
12	Click on **Next** twice. **Response:** The dialog for setting the workgroup is opened.

Setting the Workgroup

The procedure for setting the workgroup is as follows:

Step	Action
1	Select the **Workgroup** option and enter the `WORKGROUP` name. Click on **Next**.
2	Click on **Finish**. **Response:** The **Network Settings Changes** dialog is opened.
3	Click on **Yes** to restart. **Response:** Your PC now simulates a TCP/IP network and the 32-bit simulator PLCSIM may be used.

Concept Security

24

Overview

This chapter describes Concept Security.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
General Description of Concept Security	758
Access Rights	760
Changing Passwords	767
Activating Access Rights	769
Protecting Projects/DFBs	770

General Description of Concept Security

At a Glance

You can define access rights *(see page 760)* (user definitions) using Concept Security. The access rights limit the functionality of Concept and its utilities for certain users.

NOTE: The Editor LL984 cannot be protected with Concept Security.

Projects/DFBs can be protected *(see page 770)* from being edited using Concept Security.

Scope

The access rights defined for a user are valid for all projects within the Concept installation. If a user edits projects in different Concept installations, he has to be defined as a user in each Concept installation.

Max. number of users

A maximum of 128 users can be defined.

Activating Concept Security

After Concept is installed, Concept Security is inactive and must be activated by the system administrator (Supervisor).

The system administrator

Access rights are defined and Concept Security is switched on/off by the system administrator (user name: **Supervisor**).

When Concept is installed, a password file is automatically created for the "Supervisor" (system administrator) with an empty password. This user has "Supervisor" access rights.

Changing the access rights online

Concept Security and Concept/Concept-DFB can be started at the same time, i.e. the access rights can be changed while Concept/Concept-DFB is running and become active immediately.

Creating a log

In Concept, if you go into the **Options** →**Preferences** →**Common...** →**Common Preferences** dialog box in the **Logging** area and activate the **File** option (and enter a path name), the log function is activated. A file with the name YEARMONTHDAY.LOG (e.g. 19980926.LOG) is created in the folder you selected, which contains a log of all system critical (runtime relevant) changes.

The following data (and other data) is logged in the ASCII file:

- Section name
- EFB/DFB instance name, FB type name
- Pin name
- [variable name] [literal] [address]
- Old value
- New value
- User name (if password protection is activated in Concept Security)
- Date and time (also see **Options** →**Preferences** →**Common...**)

The following logging can be carried out during log-on:

- Modification of the user rights
- Deleted user
- Aborted log-on

In Concept, you can view the current log using the menu command **File** →**View Logfile**.

Encrypt Logfile

Logging write access on the PLC can be stored in an encrypted YEARMONTHDAY.ENC file (e.g. 20021025.ENC). To do this, go to the **Project Properties** (main menu **Project**) dialog box and activate the control box **Secure Application**. In Concept, you can view the current log using the menu command **File** →**View Logfile**. If the current log is encrypted, the content of the ENC file is automatically opened in a view tool and can viewed or printed there. Supervisor rights are required to do this.

Access Rights

At a glance

The access rights are set up in a hierarchy; if the user has the rights for a certain level, he also has the rights to all lower levels.

Access Right Levels

The following levels are defined (from lowest to highest):

Level	Access rights	Assigned Functionality
1	Read only	The user can view projects offline and online, but cannot change them. The user can establish a connection between the programming device and PLC and casn view variables online.
2	Reset SFC	The same functionality as above and also: Animation panel can be use for control (e.g. disable steps, disable transitions, force steps, etc.).
3	Change data	The same functionality as above and also: The user can change literals online.
4	Force data	The same functionality as above and also: Forcing variables.
5	Download	The same functionality as above and also: The user can download the program to the PLC. **Note:** To download the configuration, you at least need the access rights **Change configuration**.
6	Change program	The same functionality as above and also: The user can make any changes to the program, but not to DFBs or EFBs.
7	Change configuration	The same functionality as above and also: The user can change the PLC configuration.
8	Tools	The same functionality as above and also: The user can use Concept DFB, Concept EFB and Concept Converter.
9	Supervisor	The same functionality as above and also: The user can use Concept Security in Supervisor mode (set up users, activate/deactivate Concept Security).

Access Rights for the Main Menu File

The following table shows the minimum access rights required in Concept for the menu commands in the main menu **File**:

Menu commands in the main menu File	**Minimum access rights needed**
New Project	Change program
Open / Close	Read only
Open / Close (replacing/deleting EFBs/DFBs; error messages: FFB does not exist; FFB formula parameter was changed, DFB was changed internally)	Change program
Save project	Change data
Save project as....	Change data
Optimize project...	Change program
New section...	Change program
Open section...	Read only
Delete section...	Change program
Section properties... (read)	Read only
Section properties... (write)	Change program
Section Memory	Read only
Import...	Change program
Export...	Read only
Print...	Read only
Printer setup...	Read only
Exit	Read only

Access Rights for the Main Menu Edit

The following table shows the minimum access rights required in Concept for the menu commands in the main menu **Edit**:

Menu commands in the main menu Edit	Minimum access rights needed
Undo: Delete	Change program
Cut	Change program
Copy	Read only
Insert	Change program
Delete	Change program
Select all	Read only
Deselect all	Read only
Goto line... (text languages)	Read only
Goto counterpart (text languages)	Read only
Expand statement (text languages)	Change program
Lookup variables (text languages)	Change program
Search... (text languages)	Read only
Find Next (text languages)	Read only
Replace... (text languages)	Change program
Insert text file... (text languages)	Change program
Save as text file... (text languages)	Read only
Open Column (LL984 Editor)	Read only
Open Row (LL984 Editor)	Read only
Close Column (LL984 Editor)	Read only
Close Row (LL984 Editor)	Read only
DX Zoom... (LL984 Editor)	Read only
Reference Zoom (LL984 Editor)	Read only
Offset References... (LL984 Editor)	Read only
Replace References (LL984 Editor)	Read only

Access Rights for the Main Menu View

The following table shows the minimum access rights required in Concept for the menu commands in the main menu **View** (only for FBD, LD and SFC):

Menu commands in the main menu View	**Minimum access rights needed**
Overview	Read only
Normal	Read only
Expanded	Read only
Zoom in	Read only
Zoom out	Read only
Grid	Read only
Page breaks	Read only

Access Rights for the Main Menu Objects

The following table shows the minimum access rights required in Concept for the menu commands in the main menu **Objects**:

Menu commands in the main menu Objects	**Minimum access rights needed**
Properties (read) (only for FBD, LD and SFC)	Read only
Properties (write) (only for FBD, LD and SFC)	Change program
Select	Read only
Text	Change program
Replace variables...	Change program
Link	Change program
Vertical Link (LD Editor)	Change program
FFB: Last Type (FBD, LD Editor)	Change program
Invert input/output (FBD, LD Editor)	Change program
Insert Macro... (FBD Editor)	Change program
FFB selection... (FBD, LD Editor)	Change program
Replace FFBs... (FBD, LD Editor)	Change program
FFB Show execution order (FBD Editor)	Read only
Reverse FFB execution order (FBD Editor)	Change program
Insert contacts, coils (LD Editor)	Change program
Select column structure (SFC Editor)	Change program
Select row structure (SFC Editor)	Change program
Insert steps, transitions (SFC Editor)	Change program

Menu commands in the main menu Objects	**Minimum access rights needed**
Insert FFB, **Download**, **Save** etc. (IL Editor)	Change program
Insert FFB, **Assignment**, **Operators**, **Declaration** etc. (ST Editor)	Change program
Coils, **Insert contacts** (LL984 Editor)	Change program

Access Rights for the Main Menu Project

The following table shows the minimum access rights required in Concept for the menu commands in the main menu **Project**:

Menu commands in the main menu Project	**Minimum access rights needed**
Properties (write)	Change program
Memory Prediction	Read only
PLC configuration	Change configuration
Project Browser (write)	Change program
Execution order... (write)	Change program
Variable declarations... (write)	Change program
ASCII Messages	Read only
Search...	Read only
Trace	Read only
Find Next	Read only
Search Results	Read only
Used references...	Read only
Analyze section	Read only
Analyze program	Read only
Synchronize versions of nested DFBs	Read only
Code generation options...	Supervisor

Access Rights for the Main Menu Online

The following table shows the minimum access rights required in Concept for the menu commands in the main menu **Online**:

Menu commands in the main menu Online	Minimum access rights needed
Connect... (view only)	Read only
Connect... (change data)	Reset SFC
Connect... (change program)	Download
Connect... (change configuration)	Download
Disconnect...	Read only
Online control panel... (all commands)	Download
Single sweep trigger	Download
Controller status......	Read only
Online events...	Read only
Online diagnostics (read)	Read only
Online diagnostics (acknowledge entries)	Change data
Record changes	Change program
Object information...	Read only
Memory statistics...	Read only
Download... (IEC Program, 984 Ladder Logic, ASCII Messages, Status-RAM, Extended Memory)	Download
Download... (configuration)	Change configuration
Download changes...	Change program
Upload... (Status-RAM, Extended Memory)	Change data
Upload... (IEC Program, 984 Ladder Logic, ASCII Messages, Status-RAM)	Change program
Upload... (configuration)	Change configuration
Reference Data Editor (read only)	Read only
Reference Data Editor (write)	Change data
Reference Data Editor (force)	Force data
Disabled discretes...	Change data
Activate animation	Read only
Change literals during animation	Change data
Animation Panel (SFC Editor)	SFC Animation Panel
Animation Panel (forcing SFC steps)	SFC Animation Panel
Animation Panel (Resetting an SFC string)	SFC Animation Panel

Menu commands in the main menu Online	Minimum access rights needed
Save animation (IL, ST Editor)	Read only
Restore animation (IL, ST Editor)	Read only
Direct-mode 984LL Editor... (LL984 Editor)	Read only
power flow (LL984 Editor)	Read only
Power flow with contact state (LL984 Editor)	Read only
Trace (LL984 Editor)	Read only
ReTrace (LL984 Editor)	Read only

Access Rights for the Main Menu Options

The following table shows the minimum access rights required in Concept for the menu commands in the main menu **Options**:

Menu commands in the main menu Options	Minimum access rights needed
Confirmations...	Change program
Preferences →Common...	Change program
Preferences →Graphical Editor...	Change program
Preferences →Analysis...	Change program
Preferences →IEC Extensions...	Change program
Save settings	Change program
Save settings on close	Change program

Access Rights for the Main Menu Window

The following table shows the minimum access rights required in Concept for the menu commands in the main menu **Window**:

Menu commands in the main menu Window	Minimum access rights needed
Cascade	Read only
Slit window	Read only
Tile	Read only
Arrange icons	Read only
Close all	Read only
Messages	Read only
Name of Open Sections	Read only

Changing Passwords

Introduction

This section describes the steps necessary to change the system administrator's password and enter new users.

Changing the System Administrator's password

The following steps are only necessary when you start Concept Security for the first time after installing Concept. They describe the procedure for changing the system administrator's password:

Step	Action
1	Start access management by double clicking on the Concept Security icon.
2	Enter the user name for the `supervisor` and confirm with **OK**. Entering a password is not necessary in this case.
3	Press the **Change Password** command button.
4	Enter a password in the **Password** text box. **Note:** The password is context sensitive.
5	To confirm the password, enter the same password in the **Confirm New Password** text box. **Reaction:** If the two entries are identical, the command button **OK** is enabled.
6	Confirm the change by pressing the **OK** button
7	Exit access management with the command button **Exit**.

Entry for a user and the access rights

To enter users, assign access rights and activate Concept Security, follow these steps:

Step	Action
1	Start access management by double clicking on the Concept Security icon.
2	Enter a user name with supervisor access rights, enter the password and confirm with **OK**.
3	Select the **User** tab.
4	Press the **Add** command button.
5	Enter the user name (at least 2, maximum 16 characters) and confirm with **OK**.
6	In the **Access Rights:** list box, select the desired access rights and confirm with the command button **OK**.
7	Exit access management with the command button **Exit**.
8	To change the password for the new user, follow the procedure Changing the System Administrator's password. Enter the user name for the user that was just defined.

Activating Access Rights

Activating access rights

To activate access rights, follow these steps:

Step	Action
1	Start access management by double clicking on the Concept Security symbol.
2	Enter a user name with supervisor access rights, enter the password and acknowledge with **OK**.
3	Select the register **Options**.
4	Activate the check box **Password Required**.
5	Exit access management with the command button **Exit**. **Reaction:** Concept, Concept DFB, Concept EFB, etc. can only be started by authorized users and with the access rights defined for them.

Protecting Projects/DFBs

Introduction

With Concept Security, you can protect projects/DFBs from being changed. Protected projects can only be loaded on the PLC but cannot be changed. Protected DFBs can only be used and cannot be changed.

Protecting projects/DFBs

To protect projects/DFBs, follow these steps:

Step	Action
1	Start access management by double clicking on the Concept Security symbol.
2	Enter a user name with supervisor access rights, enter the password and confirm with **OK**.
3	Select the **Protect** register.
4	Press the command button **Select** and select the project/DFB to be protected. Confirm with **OK**. **Reaction:** The selected project/DFB will appear in a list box.
5	Select the project/DFB in the list box and press **Protect**. **Reaction:** The dialog box **Enter Password** is opened.
6	Enter a password for **Password** and acknowledge it by entering the same password for **Confirm Password**. Press **OK**. **Reaction:** The project/DFB is now protected. This is identified by a **(c)** in the list box.
7	In order to locate protected projects/DFBs quickly, it is advisable to save the list in the **Program/DFB** list box using **Save list...** .

Deactivate protection for projects/DFBs

To deactivate protection for projects/DFBs, follow these steps:

Step	Action
1	Start access management by double clicking on the Concept Security symbol.
2	Enter a user name with supervisor access rights, enter the password and confirm with **OK**.
3	Select the **Protect** register.
4	Press the command button **Select** and select the protected project/DFB that should have protection deactivated. Confirm with **OK**. **Reaction:** The selected project/DFB will appear in a list box. **or** Use **Load list...** to load a previously saved list. **Reaction:** All projects/DFBs in the loaded list will appear in the list box.
5	Select the project/DFB from the list box (identified by **(c)**) and press **Unprotect**. **Reaction:** The **Enter Password** dialog box is opened.
6	Enter the password for **Password** and press **OK**. **Reaction:** The project/DFB is no longer protected. This is identified by the **(c)** not being shown in the list box.

Appendices

Overview

Additional information that is not necessarily required for an understanding of the documentation.

What's in this Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	Tables of PLC-dependent Performance Attributes	775
B	Windows interface	799
C	List of symbols and short cut keys	819
D	IEC conformity	849
E	Configuration examples	877
F	Convert Projects/DFBs/Macros	991
G	Concept ModConnect	995
H	Convertion of Modsoft Programs	1001
I	Modsoft and 984 References	1007
J	Presettings when using Modbus Plus for startup	1011
K	Presettings when using Modbus for startup	1025
L	Startup when using Modbus with the EXECLoader	1031
M	Startup when using Modbus with DOS Loader	1049
N	Startup when using Modbus Plus with the EXECLoader	1063
O	Startup when using Modbus Plus with DOS Loader	1085
P	EXEC files	1101
Q	INI Files	1105
R	Interrupt Processing	1125
S	Automatic Connection to the PLC	1147

Tables of PLC-dependent Performance Attributes

A

Overview

The performance attributes of the different hardware platforms (Quantum, Compact, Momentum and Atrium) can be found in the following tables.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Performance of Quantum	776
Performance Attributes of Compact	781
Performance Attributes of Momentum	786
Performance Attributes of Atrium	792

Performance of Quantum

IEC and LL984 Support

Availability of IEC and LL984 support:

Performance	CPU type					
	113 02	113 03	213 04	424 0x	434 12	534 14
LL984 only	x	-	-	-	-	-
IEC only (Stripped Exec)	x	x	x	-	-	-
IEC and LL984	-	x	x	x	x	x
x = available - = not available						

Special Performance Attributes

Availability of special performance attributes:

Performance	CPU type					
	113 02	113 03	213 04	424 0x	434 12	534 14
LL984 Hot Standby	x	x	x	x	x	x
IEC Hot Standby	-	-	-	-	x	x
Interrupt processing with HLI (LL984 only)	x	x	x	x	x	x
Split memory (LL984 only with separate software	-	-	-	-	-	-
Support for XMIT loadable (LL984 only)	x	x	x	x	x	x
Support for XMIT EFB (IEC only)	-	-	-	-	-	-
Support for XXMIT EFB (IEC only)	x	x	x	x	x	x
Upload of the user program	x	x	x	x	x	x
Support of the Modbus function codes 42 (IEC only)	x	x	x	x	x	x
Password protection of connection structure with PLC	-	-	-	-	-	-
PCMCIA support	-	-	-	-	-	-

Performance	**CPU type**					
	113 02	**113 03**	**213 04**	**424 0x**	**434 12**	**534 14**
Flash memory for program and configuration	-	-	-	-	x	x
Remote Terminal Unit (RTU) configuration extension	-	-	-	-	-	-
Profibus DP configuration extension	x	x	x	x	x	x
Cyclical data exchange for configuration extension	x	x	x	x	x	x
Code generation options: Include diagnosis information	x	x	x	x	x	x
Code generation options: Fastest code	-	-	x	x	x	x
MMS Ethernet configuration extension	x	x	x	x	x	x
ASCII Messages	x	x	x	x	x	x
Peer Cop	x	x	x	x	x	x
RIO (Remote I/O)	x	x	x	x	x	x
DIO (Distributed I/O)	x	x	x	x	x	x
SYMAX I/O	x	x	x	x	x	x
800 I/O	x	x	x	x	x	x
LonWorks	x	x	x	x	x	x
A120 I/O	-	-	-	-	-	-
x = available - = not available						

Buses

Availability of the buses:

Performance	CPU type					
	113 02	113 03	213 04	424 0x	434 12	534 14
Modbus	x	x	x	x	x	x
Modbus Plus	x	x	x	x	x	x
Ethernet (TCP/IP)	x	x	x	x	x	x
Ethernet (SY/MAX)	x	x	x	x	x	x
Interbus	x	x	x	x	x	x
Interbus: PCP loadable (LL984 only)	x	x	x	x	x	x
Interbus: PCP-EFB (IEC only)	x	x	x	x	-	-
INTERBUS G4 (Generic Bus)	-	x	x	-	x	x
LonWorks (Echelon)	using NOL 911 xx and LL984	using NOL 911 xx and LL984	using NOL 911 xx and LL984	using NOL 911 xx and LL984	using NOL 911 xx and LL984	using NOL 911 xx and LL984
MVB (MultiVehicleBus)	-	-	-	-	-	-
x = available - = not available						

Block Libraries

Availability of the block libraries:

Performance	CPU type					
	113 02	113 03	213 04	424 0x	434 12	534 14
AKFEFB (IEC only)	x	x	x	x	x	x
ANA_IO (IEC only)	x	x	x	x	x	x
COMM (IEC only)	x	x	x	x	x	x
CONT_CTL (IEC only)	x	x	x	x	x	x
DIAGNO (IEC only)	x	x	x	x	x	x
EXPERTS (IEC only)	x	x	x	x	x	x
EXTENDED (IEC only)	x	x	x	x	x	x
FUZZY (IEC only)	x	x	x	x	x	x
HANDTABLEAU (IEC only)	x	x	x	x	x	x
IEC (IEC only	x	x	x	x	x	x
LIB984 (IEC only)	x	x	x	x	x	x
SYSTEM (IEC only)	x	x	x	x	x	x
LL984 (LL984 only)	x	x	x	x	x	x
x = available - = not available						

Utilities

Availability of utilities:

Performance	CPU type					
	113 02	113 03	213 04	424 0x	434 12	534 14
Concept DFB	x	x	x	x	x	x
Concept EFB	x	x	x	x	x	x
Concept SIM	x	x	x	x	x	x
Concept PLCSIM32	x	x	x	x	x	x
Concept security	x	x	x	x	x	x
Concept EXECLoader	x	x	x	x	x	x
Concept-Converter	x	x	x	x	x	x
Modsoft converter	x	x	x	x	x	x
ModConnect tool	x	x	x	x	x	x
x = available - = not available						

Runtime System

Runtime System

Performance	CPU type					
	113 02	113 03	213 04	424 0x	434 12	534 14
16 bit CPU	x	x	x	x	-	-
32 bit CPU	-	-	-	-	x	x
x = available - = not available						

Available Memory for User Program

Available memory for user program

Performance	CPU type					
	113 02	113 03	213 04	424 0x	434 12	534 14
IEC only runtime system	125k	375k	612k	-	-	-
IEC and LL984 runtime system	-	160k	330k	460k	800k	2500k
LL984 only runtime system	-	-	-	-	-	-
x = available - = not available						

Different Performance Attributes

Availability of different performance attributes:

Performance	CPU type					
	113 02	113 03	213 04	424 0x	534 14	534 14
Battery adapter required for backing up IEC programs	-	-	-	-	-	-
Floating point processor	-	-	x	x	x	x
Floating point emulation (IEC)	x	x	-	-	-	-
x = available - = not available						

Performance Attributes of Compact

IEC and LL984 Support

Availability of IEC and LL984 support:

Performance	CPU type			
	258 (512k)	265 (512k)	275 (512k)	285 (1M)
LL984 only	-	-	-	-
IEC only (Stripped Exec)	-	-	-	-
IEC and LL984	x	x	x	x
x = available - = not available				

Special Performance Attributes

Availability of special performance attributes:

Performance	CPU type			
	258 (512k)	265 (512k)	275 (512k)	285 (1M)
LL984 Hot Standby	-	-	-	-
IEC Hot Standby	-	-	-	-
Interrupt processing with HLI (LL984 only)	-	-	-	-
Split memory (LL984 only with separate software	x	x	x	x
Support for XMIT loadable (LL984 only)	x	x	x	x
Support for XMIT EFB (IEC only)	-	-	-	-
Support for XXMIT EFB (IEC only)	x	x	x	x
Upload of the user program	x	x	x	x
Support of Modbus function code 42 (IEC only)	x	x	x	x
Password protection of connection structure with PLC	x	x	x	x
PCMCIA support	-	-	x	x
Flash memory for program and configuration	x	x	x	x

Performance	**CPU type**			
	258 (512k)	**265 (512k)**	**275 (512k)**	**285 (1M)**
Remote Terminal Unit (RTU) configuration extension	x	x	x	x
Profibus DP configuration extension	-	-	-	-
Cyclical data exchange for configuration extension	-	-	-	-
Code generation options: Include diagnosis information	x	x	x	x
Code generation options: Fastest code	x	x	x	x
MMS Ethernet configuration extension	-	-	-	-
ASCII Messages	-	-	-	-
Peer Cop	-	x	x	x
RIO (Remote I/O)	-	-	-	-
DIO (Distributed I/O)	-	-	-	-
SYMAX I/O	-	-	-	-
800 I/O	-	-	-	-
LonWorks	-	-	-	-
A120 I/O	x	x	x	x
x = available - = not available				

Buses

Availability of the buses:

Performance	CPU type			
	258 (512k)	265 (512k)	275 (512k)	285 (1M)
Modbus	x	x	x	x
Modbus Plus	using BridgeModule	x	x	x
Ethernet (TCP/IP)	using BridgeModule	using BridgeModule	using BridgeModule	using BridgeModule
Ethernet (SY/MAX)	-	-	-	-
Interbus	using BKF xxx	using BKF xxx	using BKF xxx	using BKF xxx
Interbus: PCP loadable (LL984 only)	-	-	-	-
Interbus: PCP-EFB (IEC only)	-	-	-	-
LonWorks (Echelon)	-	-	-	-
MVB (MultiVehicleBus)	x	x	x	x
x = available - = not available				

Block Libraries

Availability of block libraries:

Performance	CPU type			
	258 (512k)	265 (512k)	275 (512k)	285 (1M)
AKFEFB (IEC only)	x	x	x	x
ANA_IO (IEC only)	x	x	x	x
COMM (IEC only)	-	x	x	x
CONT_CTL (IEC only)	x	x	x	x
DIAGNO (IEC only)	x	x	x	x
EXPERTS (IEC only)	x	x	x	x
EXTENDED (IEC only)	x	x	x	x
FUZZY (IEC only)	x	x	x	x
HANDTABLEAU (IEC only)	x	x	x	x
IEC (IEC only)	x	x	x	x
LIB984 (IEC only)	x	x	x	x
SYSTEM (IEC only)	x	x	x	x
LL984 (LL984 only)	x	x	x	x
x = available - = not available				

Utilities

Availability of utilities:

Performance	CPU type			
	258 (512k)	265 (512k)	275 (512k)	285 (1M)
Concept DFB	x	x	x	x
Concept EFB	x	x	x	x
Concept SIM	x	x	x	x
Concept PLCSIM32	x	x	x	x
Concept Security	x	x	x	x
Concept EXECLoader	x	x	x	x
Concept-Converter	x	x	x	x
Modsoft converter	x	x	x	x
Concept-ModConnect	-	-	-	-
x = available - = not available				

Runtime System

Runtime system

Performance	CPU type			
	258 (512k)	265 (512k)	275 (512k)	285 (1M)
16 bit CPU	-	-	-	-
32 bit CPU	x	x	x	x
x = available - = not available				

Different Performance Attributes

Availability of different performance attributes:

Performance	CPU type			
	258 (512k)	265 (512k)	275 (512k)	285 (1M)
Battery adapter required for backing up IEC programs	-	-	-	-
Floating point processing	-	-	-	-
Floating point emulation	x	x	x	x
x = available - = not available				

Performance Attributes of Momentum

IEC and LL984 Support

Availability of IEC and LL984 support:

Performance	CPU type					
	700 00 **700 10** **780 00**	**760 00**	**760 10** **780 10**	**960 20** **980 20**	**960 30** **980 30**	**970 30**
LL984 only	x	x	x	x	x	x
IEC only	-	x	x	-	x	x
IEC and LL984	-	-	-	-	-	-
x = available - = not available						

Special Performance Attributes

Availability of special performance attributes:

Performance	CPU type					
	700 00 **700 10** **780 00**	**760 00**	**760 10** **780 10**	**960 20** **980 20**	**960 30** **980 30**	**970 30**
LL984 Hot Standby	-	-	-	-	-	-
IEC Hot Standby	-	-	-	-	-	-
Interrupt processing with HLI (LL984 only)	-	-	-	-	-	-
Split memory (LL984 only with separate software	-	-	-	-	-	-
Support for the XMIT blocks (LL984 only)	x	x	x	x	x	x
Support for XMIT EFB (IEC only)	-	-	-	-	-	-
Support for XXMIT EFB (IEC only)	x	x	x	x	x	x
Uploading the user program	x	x	x	x	x	x
Support of Modbus function code 42 (IEC only)	-	x	x	-	x	x
Password protection of connection structure with PLC	-	-	-	x	x	x
PCMCIA support	-	-	-	-	-	-
Flash memory for program and configuration (LL984)	x	x	x	x	x	x
Flash memory for program and configuration (IEC)	-	-	x	-	x	x

Performance	**CPU type**					
	700 00 **700 10** **780 00**	**760 00**	**760 10** **780 10**	**960 20** **980 20**	**960 30** **980 30**	**970 30**
Remote Terminal Unit (RTU) configuration extension	-	-	-	-	-	-
Profibus DP configuration extension	-	-	-	-	-	-
Cyclical data exchange for configuration extension	-	-	-	-	-	-
Code generation options: Include diagnostics information	-	-	-	-	-	-
Code generation options: Fastest code	-	-	-	-	-	-
MMS Ethernet configuration extension	-	-	-	-	-	-
ASCII messages	-	-	-	-	-	-
Peer Cop	x	x	x	x	x	-
RIO (Remote I/O)	-	-	-	-	-	-
DIO (Distributed I/O)	-	-	-	-	-	-
TIO (Terminal I/O	x	x	x	x	x	-
SY/MAX I/O	-	-	-	-	-	-
800 I/O	-	-	-	-	-	-
LonWorks	-	-	-	-	-	-
A120 I/O	-	-	-	-	-	-
x = available - = not available						

Buses

Availability of the buses:

Performance	**CPU type**					
	700 00 **700 10** **780 00**	**760 00**	**760 10** **780 10**	**960 20** **980 20**	**960 30** **980 30**	**970 30**
Modbus (with ring card)	x	x	x	x	x	-
Modbus Plus (with ring card)	x	x	x	x	x	-
Ethernet (TCP/IP)	-	-	-	x (LL984 only)	x	x
Ethernet (SY/MAX)	-	-	-	-	-	-
INTERBUS	x	x	x	x	x	-
INTERBUS: PCP loadable (LL984 only)	-	-	-	-	-	-
INTERBUS: PCP-EFB (IEC only)	-	-	-	-	-	-
LonWorks (Echelon)	-	-	-	-	-	-
MVB (MultiVehicleBus)	-	-	-	-	-	-
x = available - = not available						

Block Libraries

Availability of the block libraries:

Performance	CPU type					
	700 00 **700 10** **780 00**	**760 00**	**760 10** **780 10**	**960 20** **980 20**	**960 30** **980 30**	**970 30**
AKFEFB (IEC only)	-	x	x	-	x	x
ANA_IO (IEC only)	-	x	x	-	x	x
COMM (IEC only)	-	-	-	-	x	x
CONT_CTL (IEC only)	-	x	x	-	x	x
DIAGNO (IEC only)	-	x	x	-	x	x
EXPERTS (IEC only)	-	-	-	-	x	x
EXTENDED (IEC only)	-	x	x	-	x	x
FUZZY (IEC only)	-	x	x	-	x	x
HANDTABLEAU (IEC only)	-	-	-	-	x	x
IEC (IEC only)	-	x	x	-	x	x
LIB984 (IEC only)	-	x	x	-	x	x
SYSTEM (IEC only)	-	x	x	-	x	x
LL984 (LL984 only)	x	x	x	x	x	x
x = available - = not available						

Utilities

Availability of utilities:

Performance	**CPU type**					
	700 00 **700 10** **780 00**	**760 00**	**760 10** **780 10**	**960 20** **980 20**	**960 30** **980 30**	**970 30**
Concept DFB	-	x	x	-	x	x
Concept EFB	-	x	x	-	x	x
Concept SIM	-	x	x	-	x	x
Concept PLCSIM32	-	x	x	-	x	x
Concept security	-	x	x	-	x	x
Concept EXECLoader	x	x	x	x	x	x
Concept-Converter	x	x	x	x	x	x
Modsoft converter	x	x	x	x	x	x
Concept-ModConnect	x	x	x	x	x	x
x = available - = not available						

Runtime System

Runtime system

Performance	**CPU type**					
	700 00 **700 10** **780 00**	**760 00**	**760 10** **780 10**	**960 20** **980 20**	**960 30** **980 30**	**970 30**
16 bit CPU	x	x	x	x	x	x
32 bit CPU	-	-	-	-	-	-
x = available - = not available						

Different Performance Attributes

Availability of different performance attributes:

Performance	**CPU type**					
	700 00 **700 10** **780 00**	**760 00**	**760 10** **780 10**	**960 20** **980 20**	**960 30** **980 30**	**970 30**
Battery adapter required for backing up IEC programs	-	x	-	-	-	-
Floating point processor	-	-	-	-	-	-
Floating point emulation (IEC)	-	x	x	-	x	x
x = available - = not available						

Performance Attributes of Atrium

IEC and LL984 Support

Availability of IEC and LL984 support:

Performance	CPU type
	121 01 (2M) **241 01 (4M)** **241 11 (4M)**
LL984 only	-
IEC only (Stripped Exec)	x
IEC and LL984	-
x = available - = not available	

Special Performance Attributes

Availability of special performance attributes:

Performance	CPU type
	121 01 (2M) **241 01 (4M)** **241 11 (4M)**
LL984 Hot Standby	-
IEC Hot Standby	-
Interrupt processing with HLI (LL984 only)	-
Split memory (LL984 only with separate software	-
Support for XMIT loadable (LL984 only)	-
Support for XMIT EFB (IEC only)	-
Support for XXMIT EFB (IEC only)	-
Upload of the user program	x
Support of Modbus function code 42 (IEC only)	x
Password protection of connection structure with PLC	-
PCMCIA support	-

Performance	**CPU type**
	121 01 (2M) **241 01 (4M)** **241 11 (4M)**
Flash memory for program and configuration	-
Remote Terminal Unit (RTU) configuration extension	-
Profibus DP configuration extension	-
Cyclical data exchange for configuration extension	-
Code generation options: Include diagnosis information	-
Code generation options: Fastest code	-
MMS Ethernet configuration extension	-
ASCII Messages	-
Peer Cop	x
RIO (Remote I/O)	-
DIO (Distributed I/O)	-
SYMAX I/O	-
800 I/O	-
LonWorks	-
A120 I/O	-
x = available - = not available	

Buses

Availability of the buses:

Performance	CPU type
	121 01 (2M) **241 01 (2M)** **241 11 (4M)**
Modbus	-
Modbus Plus	x
Ethernet (TCP/IP)	-
Ethernet (SY/MAX)	-
Interbus	x x x
Interbus: PCP loadable (LL984 only)	-
Interbus: PCP-EFB (IEC only)	-
Profibus	- - -
LonWorks (Echelon)	-
MVB (MultiVehicleBus)	-
x = available - = not available	

Block Libraries

Availability of block libraries:

Performance	CPU type
	121 01 (2M) **241 01 (2M)** **241 11 (4M)**
AKFEFB (IEC only)	x
ANA_IO (IEC only)	x
COMM (IEC only)	x
CONT_CTL (IEC only)	x
DIAGNO (IEC only)	x
EXPERTS (IEC only)	x
EXTENDED (IEC only)	x
FUZZY (IEC only)	x
HANDTABLEAU (IEC only)	x
IEC (IEC only)	x
LIB984 (IEC only)	x
SYSTEM (IEC only)	x
LL984 (LL984 only)	-
x = available - = not available	

Utilities

Availability of utilities:

Performance	**CPU type**
	121 01 (2M) **241 01 (2M)** **241 11 (4M)**
Concept DFB	x
Concept EFB	x
Concept SIM	x
Concept PLCSIM32	x
Concept Security	x
Concept EXECLoader	x
Concept-Converter	x
Modsoft converter	x
Concept-ModConnect	-
x = available - = not available	

Runtime System

Runtime system

Performance	**CPU type**
	121 01 (2M) **241 01 (2M)** **241 11 (4M)**
16 bit CPU	-
32 bit CPU	x
x = available - = not available	

Different Performance Attributes

Availability of different performance attributes:

Performance	**CPU type**
	121 01 (2M) **241 01 (2M)** **241 11 (4M)**
Battery adapter required for backing up IEC programs	-
Floating point processor	- x x
Floating point emulation	x - -
x = available - = not available	

Windows interface

B

Overview

The chapter describes the most important properties of Concept’s Windows interface. Further information can be found in the Microsoft Windows manuals.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
B.1	Window	800
B.2	Menu commands	806
B.3	Dialog boxes	808
B.4	Generating a project symbol	811
B.5	Online help	813

B.1 Window

Overview

This section describes the types of windows and window elements in Windows.

What's in this Section?

This section contains the following topics:

Topic	Page
Window Types	801
Elements of a window	803

Window Types

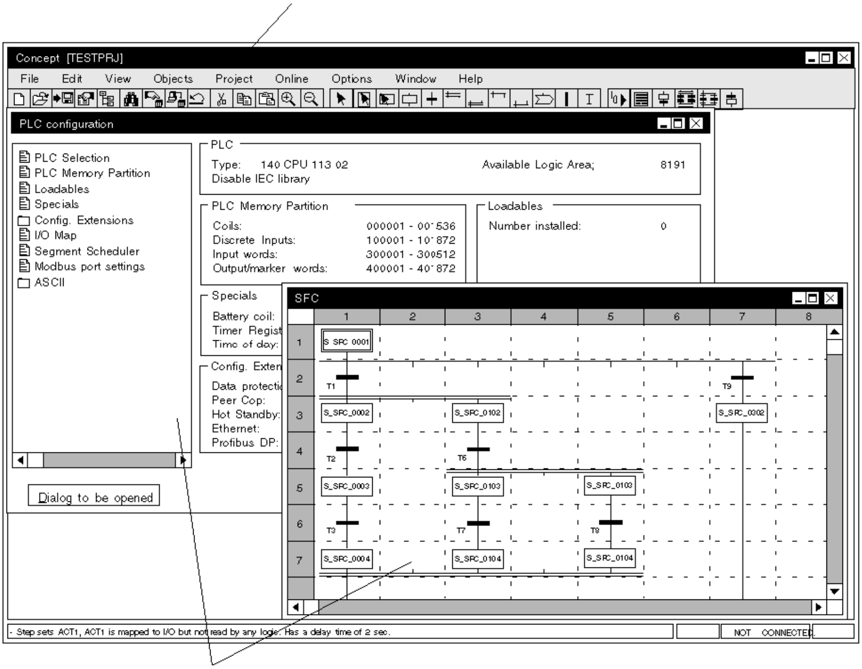
Introduction

In Windows there are two types of windows:

- Application Window
- Document Window

Types of window:

Application window (project)



PLC
Type: 140 CPU 113 02 Available Logic Area; 8191
Disable IEC library

PLC Memory Partition
Coils: 000001 - 00`536
Discrete Inputs: 100001 - 10`872
Input words: 300001 - 300512
Output/marker words: 400001 - 40`872

Loadables
Number installed: 0

Specials
Battery coil:
Timer Regist
Time of day:

Config. Exten
Data protecti
Peer Cop:
Hot Standby:
Ethernet:
Profibus DP:

- Step sets ACT1, ACT1 is mapped to I/O but not read by any logic. Has a delay time of 2 sec.

Document window (PLC configuration, section)

Application Window

When Concept is started the application window is opened on your desktop. The application window can be moved to any position on the desktop. Alternatively it can be minimized to a button on the task bar.

A project can be opened or created in this application window. The name of the project then appears in the title bar of the application window.

Document Window

After opening or creating a project you can open different document windows. Document windows are, for example, sections in which a user program is created or the document window of the PLC configuration.

Several document windows can be open simultaneously, but only one of these can be active. An active document window can be recognized by the color of the title bar.

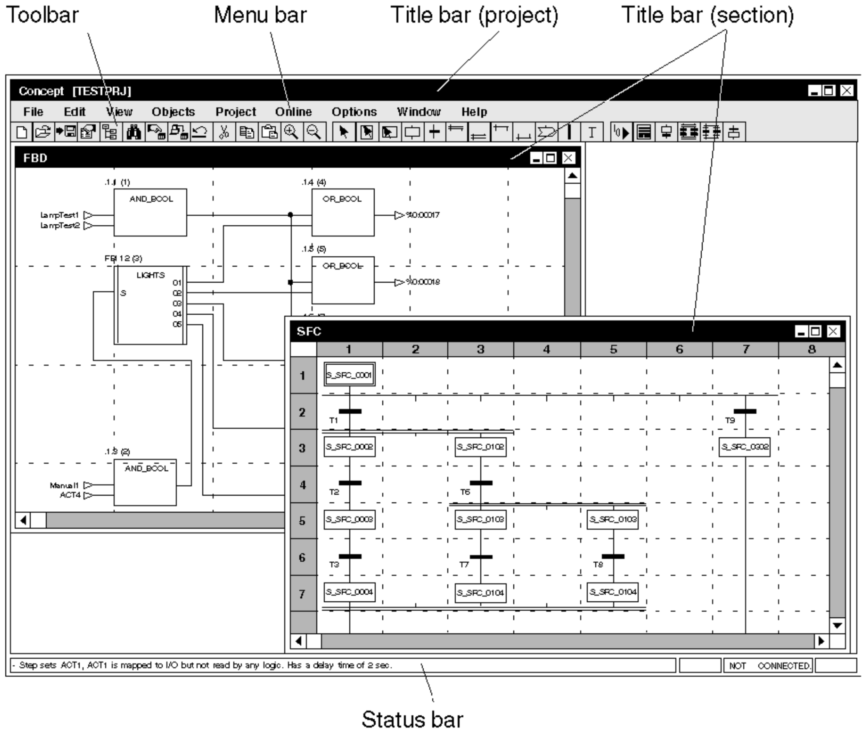
Depending on the active document window the menu commands change in the pull down menus and the tool bar of the application window.

Elements of a window

At a Glance

This section describes the Concept specific elements of a window.

Elements of a window:



- Step sets ACT1, ACT1 is mapped to I/O but not read by any logic. Has a delay time of 2 sec.

Title bar

A project’s title bar shows the name of the active application (i.e. Concept) and the name of the project. When coupled with a PLC the node address of the PLC is indicated in angled brackets (<>). If this PLC is on another network the routing path is also indicated.

If a document window (e.g. a section) is enlarged to full screen, i.e. the section takes up the entire application window, the name of the document window (e.g. the section name) appears in the title bar.

Document windows which are not enlarged to full screen have their own title bar in which the name of the document window is indicated.

Menu Bar

The menu bar of the application window contains various main menus. The contents of the menu bar depend on the active document window.

Toolbar

The toolbar consists of buttons which correspond to a menu command on the pull-down menus. The range and content of the toolbar depend on which window is active.

There are three different ways a button can be represented:

- grayed
  The command is currently unavailable. One or more other commands must be executed before the desired button can be used.
- unpressed
  The command can be selected.
- pressed
  The command is active.

Status bar

The appearance of the status bar depends on whether the project is open and the programming language used in the section.

In the first part of the status bar various information is displayed depending on the selected object.

- If a dialog box is open or a menu command or button has been selected some help will be given about it. To display the help select a menu command or a button with the left mouse button and hold it down. A short description of the menu command or button appears in the status bar. To execute the menu command/button release the mouse button. If execution of the menu command/button is not required, move the pointer away from the active area (the description in the status bar disappears) and then release the mouse button.
- If an FFB, a parameter to an input/output, a step or a transition has been selected, a comment about the selected object is displayed. With parameters and transitions the assigned direct address (only in case of located variables) is also displayed.

The second part of the status bar (status of the active section) indicates whether the section is in animation mode or the section is disabled.

- **ANIMATED**
  The section is animated.
- **INHIBITED**
  The section is inhibited and will not be processed.

The third part of the status bar indicates the status of the PLC.

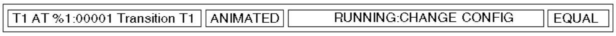
- **NOT CONNECTED.**
  The programming device is not coupled with a PLC.

- **STOPPED**
  The program on the PLC is suspended.
- **RUNNING: CHANGE CONFIG**
  The program on the PLC is running and was connected with the access **Change Configuration**.

In the fourth part of the status bar the program status between the PLC and programming device is displayed. This display only appears if a project is open and the programming device with PLC is online.

- **EQUAL**
  The program on the programming device and the PLC is consistent.
- **UNEQUAL**
  The program on the programming device and the PLC is not consistent. To establish consistency use the menu command **Online** →**Load...** .
- **MODIFIED**
  The program on the programming device was modified. The modifications can be made online in the PLC with the menu command **Online** →**Load changes**.

Status bar:



T1 AT %1:00001 Transition T1 | ANIMATED | RUNNING:CHANGE CONFIG | EQUAL

B.2 Menu commands

Menu commands

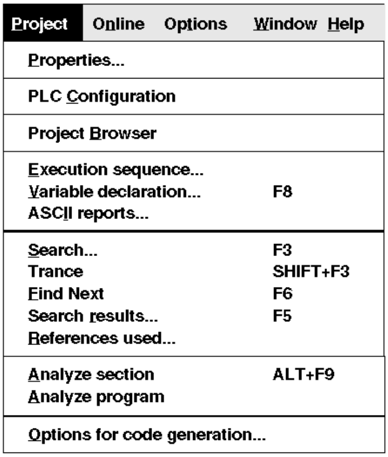
At a Glance

The titles of the individual menus are displayed in the menu bar. The menu commands are listed in the pull-down menus. As in Windows, each Concept window and dialog box has a system menu. This menu is opened using the small box in the top left-hand corner of the window.

A pull-down menu is opened by left-clicking on the title of the menu. To go directly to a menu command, drag the mouse pointer down the menu and then release the mouse button.

The menu can be closed by clicking on the title of the menu or anywhere outside of the menu.

Typical pull-down menu:



Underlined letter

A main menu (menu title) and subsequently a menu command can be selected by holding down **Alt** and simultaneously entering the underlined letter in the menu title and then that of the menu command. If, for instance, from the menu **Project** you want to execute the menu command **Search...** press **Alt**+**P** to open the menu and then **Alt**+**S** to execute the menu command.

Grayed out menu command

The command is currently unavailable. One or more other commands must be executed before the desired menu command can be executed.

Suspension points (…) after the menu command

On execution of this menu command a dialog box appears with options, which must be selected before execution.

Check mark (√) before the menu command

The menu command is active. If the menu command is selected the check mark disappears and the menu command is inactive. The check mark is mostly used to identify active modes (e.g. normal display, dial in mode etc.).

Shortcut keys

The key combinations (e.g. **F8**, **Alt**+**F9**, **Ctrl**+**R**) after the menu command are shortcut keys for executing this menu command. Using this key or key combination the menu command can be selected, without having to open the menu.

B.3 Dialog boxes

Dialog boxes

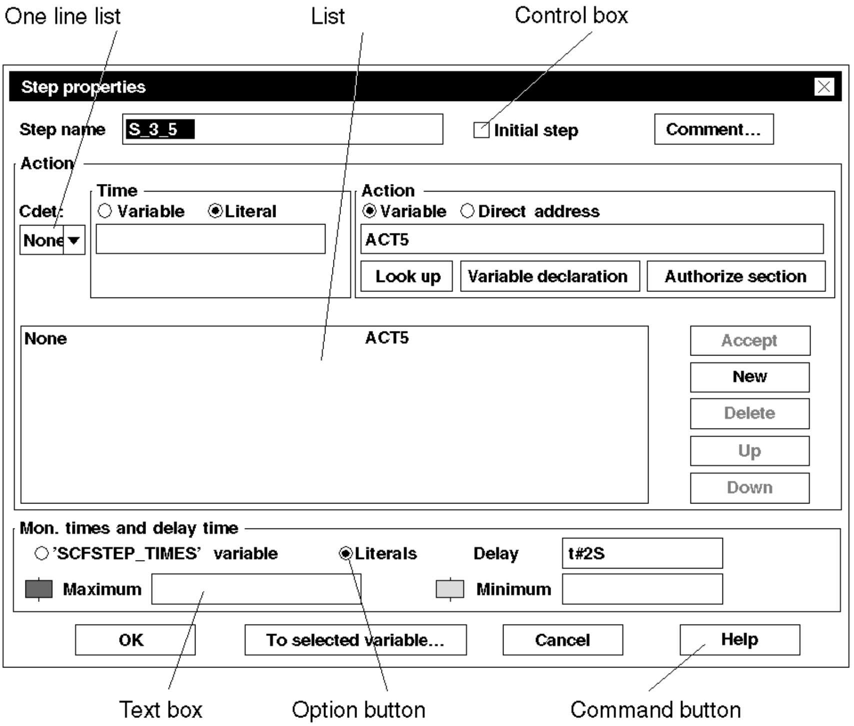
At a Glance

In Concept dialog boxes are displayed if additional information is required from you in order to perform a particular task. Potentially necessary information is also communicated in this way.

Most dialog boxes contain options which can be selected, textboxes, in which text can be entered, and buttons which can be pressed.

Grayed out options are currently not available. One or more other commands must be executed, or options selected or deselected, before the desired option can be activated.

Concept specific basics of a window:



Step properties

Step name: S_3_5

□ Initial step

Action

Cdet: None

Time: ○ Variable ● Literal

Action: ● Variable ○ Direct address

ACT5

None ACT5

Mon. times and delay time

○ 'SCFSTEP_TIMES' variable ● Literals

Delay: t#2S

Maximum

Minimum

Command buttons

Command buttons are used to initiate actions immediately, e.g. executing or aborting a command. Command buttons include e.g. **OK**, **Abort** and **Help**.

Command buttons followed by suspension points (…), open a further dialog box. A command button with a "greater than" sign (>>) extends the active dialog box.

The standard setting is identified by a dark margin. This command button can be selected by pressing **Enter**.

To close a dialog box without executing a command select the command button **Cancel**.

Text boxes

Information (text) is entered into a text box.

If you enter an empty text box an insertion point appears in the far left of the box. The entered text begins at this insertion point. If text is already present within the respective box, it will be selected and replaced by the new text automatically. The text can, however, also be deleted by pressing **Delete** or **Backspace**.

Lists

In a list the available selection possibilities are listed. If more possibilities are available than fit into the list, the scrollbar or the arrow keys can be used to move within the list.

As a rule only a single entry can be chosen form the list. There are, however, some cases in which several entries can be chosen, e.g. when opening sections.

One line lists

A single line list box initially appears as a rectangular box, in which the current selection (the default value) is selected. If the arrow in the right of the box is selected, a list of the available selection possibilities opens. If more possibilities are available than fit into the list, then the scrollbar or arrow keys can be used to move around the list.

Option buttons

Option buttons represent mutually exclusive options. In each case only one option can be chosen.

The selected option button is identified by a black dot.

If the option name contains an underlined letter, the option button can be activated from any position in the dialog box by holding down **Alt** and entering the underlined letter.

Check box

A check box next to an option means that the option can be activated or deactivated. Any number of check box options can be activated.

Activated options are identified by an X or a check mark (√).

If the option name contains an underlined letter, the check box can be activated or deactivated from any position in the dialog box by holding down **Alt** and entering the underlined letter.

B.4 Generating a project symbol

Creating a Project Symbol in a Program Group

Introduction

Creating a project symbol allows you to immediately load a certain project and/or connect to a PLC when opening Concept. In this way, one or more program groups can be created, which e.g. contain all the projects in a system.

NOTE: A symbol can only be created for an existing project. Otherwise an error message appears when starting.

Creating a symbol for projects

Follow these steps to create a project symbol:

Step	Action
1	Under **Start** →**Settings** →**Taskbar...**, you can open the **Taskbar Properties** dialog box.
2	In the register **Start Menu Programs**/**Expanded** (Win2000), select the **Add...** command button.
3	In the **Create Shortcut** dialog box, select the **Browse...** command button.
4	In the **Browse** dialog box, go to the Concept installation path and double-click on the file **CONCEPT.EXE**. **Result:** The **Browse** dialog box is closed and the file **CONCEPT.EXE** is entered, including the path, in the **Command line:** text box, e.g. **C:\CONCEPT\CONCEPT.EXE**.
5	Now add the project path and project name to the command line, e.g. **C:\CONCEPT\CONCEPT.EXE PLANT1.PRJ** and confirm the entry using **Next>** command button. **Note:** To create a connection to any PLC, add additional Parameters *(see page 1148)* to the command line.
6	In the **Select program group** dialog box, select an existing program group for the symbol or create a new one using **New folder...**. Confirm the entry using the **Next>** command button.
7	In the **Select program designation** dialog box, select the project name and confirm using the **Finish** command button.
8	Close the **Taskbar Properties** dialog box with **OK**. **Result:** The properties dialog box is closed and the project symbol is available in the start menu of the folder you selected.
9	Open the folder with the project symbol in the Start Menu. Select the project symbol and click the right mouse button. **Result:** A menu window is opened.

Step	Action
10	Select the **Properties** command button. **Result:** The **"Project Symbol Name" Properties** dialog box is opened.
11	Go to the **Connection** register and complete the command line **Working directory**/**Target** (Win2000) with the name of the project directory, e.g. **C:\CONCEPT\PROJECTS**. Confirm the entry using the **Set** command button.
12	Then exit the dialog box by selecting **OK**.
13	Open the project by clicking on the project symbol.

Creating a symbol for DFBs

In this way, symbols can also be created for DFBs. To do this, select the file **CCEPTDFB.EXE** in step 4 and add the DFB name and path instead of the project name and path in step 5.

B.5 Online help

Overview

This section describes use of online help.

What's in this Section?

This section contains the following topics:

Topic	Page
At a Glance	814
How the Online Help is set out	815

At a Glance

General information

The online help is used to quickly and easily obtain information about the task being performed, the use of an unfamiliar command or the functions, Function Blocks and modules.

The online help is available throughout Concept.

NOTE: The option **Use polygon acceleration** may not be used if the graphics card has hardware acceleration functions. Use of these may still lead to the graphics in the online help being incomplete. A detailed description of how to switch off the acceleration function will be found in the graphics card's user manual.

Starting the online help

There are several methods of calling up the online help:

- Invoking the contents)
  There are two methods of invoking the online help contents:
  - To invoke the online help contents, select the menu command **Help** → **Contents**.
  - In the program group Concept open the help symbol.
- Help with the execution of a menu command
  There are two methods of invoking help with a menu command:
  - using the mouse)
    To obtain an explanation select the menu command with the left mouse button, hold down the mouse button, press **F1**, and then release the mouse button.
  - using the keyboard)
    To obtain an explanation of a menu command, select it and then press **F1**.
- Help with a dialog
  There are two methods of invoking help with a dialog:
  - To obtain an explanation of a dialog, click on the command button **Help**in the dialog itself.
  - To obtain an explanation of a dialog, press **F1**in the dialog itself.
- Help with operating an EFB
  To obtain an explanation of the operation of the EFB, click on the command button **Help with type** within the dialog with the EFB properties.
- Help with the operation of a module
  In the dialog **I/O module selection** click on the command button **Help with module**, to obtain an explanation of the operation of a module.

How the Online Help is set out

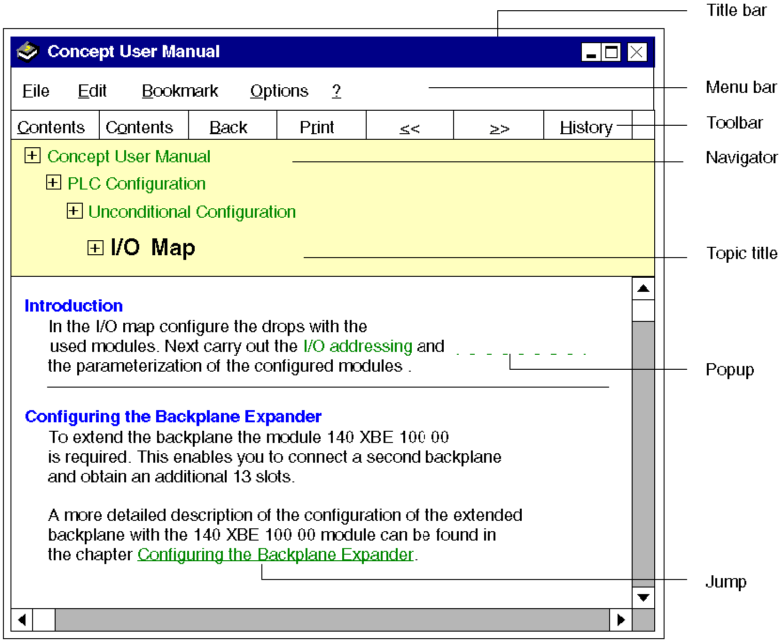
Introduction

If you start the online help, the Windows Help system opens, containing either

- a table of contents (if you started with **Help** →**Contents** or the icon),
- or containing a description of the dialog (if you started with the **Help** command button),
- or containing a description of an EFB (if you started with the **Help on Type** command button),
- or containing a description of a module (if you started with the **Module Help** command button),

This section describes the Concept specific basics of the online help window.

Online help window:



Introduction

In the I/O map configure the drops with the used modules. Next carry out the I/O addressing and the parameterization of the configured modules .

Configuring the Backplane Expander

To extend the backplane the module 140 XBE 100 00 is required. This enables you to connect a second backplane and obtain an additional 13 slots.

A more detailed description of the configuration of the extended backplane with the 140 XBE 100 00 module can be found in the chapter Configuring the Backplane Expander.

Title Bar

The title bar contains the active help file names, or in other words the help project.

Menu Bar

A description of the standard menu bar can be found in the respective Microsoft Windows manual.

Toolbar

The following buttons are available in Concept:

- **Contents**
  This button is used to invoke the online help contents directory.
  Details about this function can be found in the corresponding Windows Manual.
  Note: If you jump *(see page 817)* between different help projects and click the **Contents** button, the contents of the invoked help project (rather than the current one) is displayed. This is a Microsoft error. The Navigator is available to allow you to navigate within the current help project (related topics *Navigator, page 816*).
- **Index**
  This button is used to invoke an index for finding help texts.
  Details about this function can be found in the corresponding Windows Manual.
  Note: If you want to carry out a search of the whole text, press the **Index** command button, select the **Search** index card, choose the desired search function and type in the term you're looking for.
- **Back**
  This button is used to invoke the previously read help text.
- **Print**
  This button is used to print out the current topic (the current help topic).
- **<<**
  This button is used to "browse" the previous help text. This button is used to read the online help like a book. When you have reached the first "page" of the online help (contents directory), the button is hidden.
- **>>**
  This button is used to "browse" to the next help text. This button is used to read the online help like a book. When you have reached the last "page" of the online help, the button is hidden.
- **History**
  When you use this button a window opens which displays all of the help topics that are already open.

Title of Topic

The topic title refers to the title of a chapter from paper documentation. This topic title always remains visible, even if, in the case of long documents, the text is moved in the window.

Navigator

The Navigator is in the topic title. It serves as a navigator inside the help projects.

Jump

A jump can be recognized by the fact it is written in green and is underlined. When you click on a jump, the help text corresponding to this key word/ topic appears. Jumps correspond to "related topics" entries in paper documents, the pages are however removed for your convenience. The invoked help text is then replaced by a new help text.

Popup

A popup can be recognized by the fact it is written in green and has a dotted line under it. When you click on a popup, the help text corresponding to this key word appears. Popups correspond to glossary entries in paper documents, however, the pages here are removed for your convenience. To display the text, a popup window is opened. This popup window may contain further popups. The popup window is cleared by re-clicking on it or pressing any key. This does not replace the present help text.

List of symbols and short cut keys

C

Description

Each editor and the PLC configuration have their own list of symbols available. This facilitates access to frequently used functions. It is also possible to call up many functions with short cut keys instead of menu commands.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
C.1	Icon bar	820
C.2	Short cut keys	832

C.1 Icon bar

Description

This section describes the icon bar icons. In the icon bars there are editor independent and editor dependent icons.

What's in this Section?

This section contains the following topics:

Topic	Page
General icon bar	821
Icon bar in the FBD editor	822
Icon bar in the SFC-Editor	823
Icon bar in the LD editor	825
List of Symbols in the IL and ST Editor	827
List of Symbols in the LL984-Editor	828
Icons in PLC Configuration	829
Toolbar in the RDE Editor	830
Toolbar in the Project Browser	831

General icon bar

Symbols

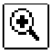
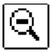
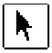
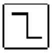
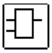
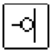
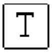
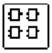
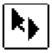
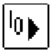
The table below shows the available symbols and their corresponding menu entry commands:

Symbol	Menu entry command executed
	File →Open...
	File →New section... / New DFB section...
	File →Open section...
	File →Save
	Project →Variable declaration...
	Project →Search…
	Online →Online control panel...
	Online →Download changes...
	Edit →Undo: Delete
	Edit →Cut
	Edit →Copy
	Edit →Paste

Icon bar in the FBD editor

Symbols

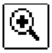
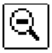
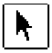
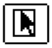
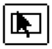
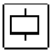
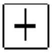
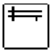
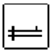
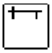
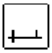
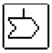
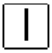
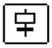
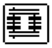
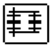
The table shows the additional icons available in the FBD editor and the corresponding menu entry commands (see also *General icon bar, page 821*):

Symbol	Menu entry command executed
	**View →Zoom in**
	**View →Zoom out**
	**Objects →Select**
	**Objects →Link**
	**Objects →FFB: Last Type**
	**Objects →Invert Input/Output**
	**Objects →Text**
	**Objects →FFB selection...**
	**Online →Animate selected**
	**Online →Animate booleans**

Icon bar in the SFC-Editor

Symbols

The table shows the additional icons available in the SFC editor and the corresponding menu entry commands (see also *General icon bar, page 821*):

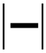
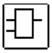
Symbol	Menu entry command executed
	**View →Zoom in**
	**View →Zoom out**
	**Objects →Select**
	**Objects →Select column structure**
	**Objects →Select row structure**
	**Objects →Step**
	**Objects →Transition**
	**Objects →Parallel branch**
	**Objects →Parallel joint**
	**Objects →Alternative branch**
	**Objects →Alternative joint**
	**Objects →Jump**
	**Objects →Link**
	**Objects →Step - Transition sequence**
	**Objects →Structured Parallel sequence**
	**Objects →Structured Alternative sequence**

Symbol	Menu entry command executed
	Objects →Transition - Step sequence
	Objects →Text
	Online →Animate
	Online →Animation Panel functions

Icon bar in the LD editor

Symbols

The table shows the additional symbols available in the LD editor and the corresponding menu entry commands (please also refer to the *General icon bar, page 821*):

Symbol	Menu entry command executed
	View →Zoom in
	View →Zoom out
	Objects →Select
	Objects →Link
	**Objects →Direct Link**
	Objects →Vertical Link
	**Objects →FFB: Last Type**
	Objects →Invert Input/Output
	Objects →Text
	Objects →FFB selection...
	Objects →Coil
	Objects →Coil - Negated
	Objects →Contact - Normally Open
	Objects →Contact – Normally Closed

Symbol	Menu entry command executed
↖▶	**Online →Animate selected**
|0▶	**Online →Animate booleans**

List of Symbols in the IL and ST Editor

Symbols

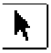
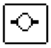
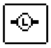
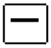
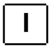
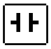
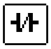
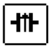
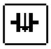
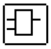
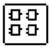
The table shows the additional symbols available in the IL and ST editor and the corresponding menu entry commands (see also *General icon bar, page 821*):

Symbol	Menu Entry Command Executed
⊞	**Objects →Insert FFB**
▶	**Online →Watch Selected**
|0▶	**Online →Animate booleans**

List of Symbols in the LL984-Editor

Symbols

The table shows the additional symbols available in the LL984 editor and the corresponding menu entry commands (see also *General icon bar, page 821*):

Symbol	Menu Entry Command Executed
	**Objects →Select**
	**Objects →Coil**
	**Objects →Coil - Retentive**
	**Objects →Horiz Short**
	**Objects →Vertical Short**
	**Objects →Contact – Normally Open**
	**Objects →Contact – Normally Closed**
	**Objects →Contact – Pos Trans**
	**Objects →Contact – Neg Trans**
	**Objects →Instruction: Last Type**
	**Objects →List Instructions...**

Icons in PLC Configuration

Icons

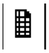
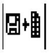
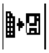
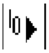
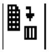
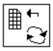
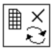
The table shows the icons also available in PLC configuration and their allocated menu commands (related topics: *General icon bar, page 821*):

Icon	Executed menu command
	PLC configuration →PLC Selection...
	PLC configuration →Memory Partitions...
	PLC configuration →ASCII Setup...
	PLC configuration →Loadables...
	PLC configuration →Config. Extension...
	PLC configuration →Segment scheduler...
	PLC configuration →I/O Map...
	PLC configuration →Data Protection...
	PLC configuration →Peer Cop...
	PLC configuration →Ethernet / I/O Scanner...
	PLC configuration →Hot Standby...
ASCII	**PLC configuration →ASCII Port Settings...**
	PLC configuration →Modbus Port Settings...
	PLC configuration →Specials...

Toolbar in the RDE Editor

Icons

The table shows the icons also available in the RDE Editor and their allocated menu commands (see also *General icon bar, page 821*):

Icon	Executed menu command
	**Template →New Template...**
	**Template →Open Template...**
	**Template →Save Template**
	**Online →Animate**
	**Online →Download Reference Data**
	**Online →Get CSL**
	**Online →Delete CSL**

Toolbar in the Project Browser

Icons

The table shows the additional symbols available in the project browser and the corresponding menu commands (also see *General icon bar, page 821*):

Icon	Menu command executed
|0▶|	**Project Shortcut Menu →Animate Enable States**
▤	**Project Shortcut Menu →Show Detailed View**

C.2 Short cut keys

Description

This section describes the available short cut keys. There are editor independent and editor dependent short cut keys.

What's in this Section?

This section contains the following topics:

Topic	Page
General Short Cut Keys	833
Short Cut Keys in the IL, ST and Data Type Editor	834
Short Cut Keys in the FBD and SFC Editor	837
Shortcut keys in the LD-Editor	841
Short Cut Keys in the LL984-Editor	847

General Short Cut Keys

Short Cut Keys

The table shows the short cut keys available and the corresponding menu entry command:

Short Cut Keys	Menu Entry Command Executed
F1	Calls the context-sensitive online help. Use this key to call up an explanation of the menu entry command or dialog chosen. In dialogs, this key corresponds to the menu entry command **Help**.
Ctrl+F4	**System menu (for the document window) →Close document window**
Ctrl+F6	**System menu (for the document window) →Next**
Ctrl+S	**File →Save project/save DFB**
Alt+F4	**File →Quit the application window (Concept-Application)**
F8	**Project →Variable declarations...**
F3	**Project →Search**
Shift+F3	**Project →Trace**
F5	**Project →Search history...**
F6	**Project →Search next**
Alt+F9	**Project →Analyze section**
Ctrl+P	**Online →Online control panel...**
F9	**Online →Single sweep trigger**
Ctrl+R	**Online →Reference Data Editor**
Shift+F5	**Window →Cascade**
Shift+F4	**Window →Tile Vertically**

Short Cut Keys in the IL, ST and Data Type Editor

Calling up menu command entries

The table shows the short cut keys available in the IL, ST and Data Type Editor and the corresponding menu entry commands (see also *General Short Cut Keys, page 833*):

Key	Menu Entry Command Executed
Ctrl+Z	**Edit →Undo delete**
Ctrl+X	**Edit →Cut**
Ctrl+C	**Edit →Copy**
Ctrl+V	**Edit →Paste**
Del	**Edit →Delete**
Ctrl+G	**Edit →Goto line...**
Ctrl+J	**Edit →Goto counterpart**
Ctrl+E	**Edit →Expand statement**
Alt+F8	**Edit →Lookup variables**
Ctrl+F	**Edit →Find next**
Ctrl+H	**Edit →Replace...**
Ctrl+Y	**Online →Animate Booleans**
Ctrl+I	**Online →Inspect Selected**
Ctrl+W	**Online →Watch Selected**

Moving insertion marks in the text

Moving insertion marks in the text:

Key	Moving
Down	Onto the next line
Up	Onto the previous line
Ctrl+**G**	Onto a specific line
End	To the end of the line
Home	To the beginning of the line
Picture up	Into the next window
Picture up	Into the previous window
Ctrl+**Right**	To the next word
Ctrl+**Left**	To the previous word
Ctrl+**End**	To the end of the document
Ctrl+**Home**	To the beginning of the document

Deleting text

Deleting text:

Key	Function
Backspace Key (Delete backwards)	Deleting a mark (or deleting marked text) to the left of the insertion mark.
Del	Deleting a character (or deleting marked text) to the right of the insertion mark.
Ctrl+**Backspace key** (Delete backwards)	Deleting a line

Marking text

Marking text:

Key	**Extending the marking**
Shift+**Right**	to the next character
Shift+**Left**	to the previous character
Ctrl+**Shift**+**Right**	to the next word
Ctrl+**Shift**+**Left**	to the previous word
Shift+**Down**	to the next line
Shift+**Up**	to the previous line
Shift+**End**	to the end of the line
Shift+**Home**	to the beginning of the line
Shift+**Picture down**	to a window underneath
Shift+**Picture up**	to a window above
Ctrl+**Shift**+**Picture down**	to the end of the current window
Ctrl+**Shift**+**Picture up**	to the beginning of the current window
Ctrl+**Shift**+**End**	to the end of the document
Ctrl+**Shift**+**Home**	to the beginning of the document

Editing text

Editing text:

Key	**Function**
Ctrl+**X**	Deleting marked text and saving in the clipboard
Ctrl+**C**	Copying marked text and saving in the clipboard
Entering the new text	Replacing marked text
Del	Deleting marked text without saving in the clipboard
Ctrl+**V**	Replacing marked text with text from the clipboard.
Ctrl+**F**	Searching for text
Ctrl+**R**	Replacing text

Short Cut Keys in the FBD and SFC Editor

At a Glance

Concept supports the work with the keyboard in the graphic editors. Although the mouse is a more appropriate input tool, it is nevertheless possible to operate Concept with the keyboard alone – especially in machine environments. The editors behave in the same way regardless of whether they are operated with the mouse or with the keyboard.

Rules

The following general rules need to be observed:

- The space bar corresponds to the left mouse button, i.e. the space bar is used for selecting and moving.
- The enter key corresponds to the double click with the left mouse button – for example, the input key is used to call up the properties dialog of objects.
- The shift key is used in conjunction with the keyboard exactly as it is with the mouse – for example, the shift key is used to extend an object selection or to reselect a few objects from a number which have already been selected.

Calling up menu command entries

The table shows the short cut keys available in the FBD and SFC editor and the corresponding menu entry commands (see also *General Short Cut Keys, page 833*):

Key	**Menu Entry Command Executed**
Ctrl+A	**Edit →Select All**
Ctrl+Z	**Edit →Undo delete**
Ctrl+X	**Edit →Cut**
Ctrl+C	**Edit →Copy**
Ctrl+V	**Edit →Paste**
Del	**Edit →Delete**
Ctrl+O	**View →Overview**
Ctrl+N	**View →Normal**
Ctrl+E	**View →Expanded** (only in SFC)
Ctrl++	**View →Zoom in**
Ctrl+-	**View →Zoom out**
Ctrl+Y	In the FBD Editor: **Online →Animate booleans** In SFC-Editor: **Online →Animate**
Ctrl+W	**Online →Animate selected** (in FBD)

Moving the cursor

Moving the cursor:

Key	Function
Cursor keys	The **cursor keys** move the cursor inside the document window. The cursor is moved further around a Pixel. If the cursor is at the edge of the document window, pressing the cursor keys again will page the document window in the corresponding direction.
Ctrl+Cursor Keys	When the **Strg** key is pressed, the **cursor keys** move the cursor inside the document window. The cursor is moved further around a logical unit (depending on the active editor). If the cursor is at the edge of the document window, pressing the **cursor keys** again will page the document window in the corresponding direction
Home	The **Pos1** key moves the cursor to the left-hand edge of the document window.
End	The **End** key moves the cursor to the right-hand edge of the document window.

Scrolling

Scrolling:

Key	Function
Ctrl+Home	When the **Ctrl** key is pressed, the **Pos1** key moves the document window to the upper left-hand corner of the section.
Ctrl+End	When the **Ctrl** key is pressed, the **End** key moves the document window to the lower right-hand corner of the section.
Picture up	The **picture up** key scrolls the document window one screen page upwards, while the cursor remains in the same position in the document window.
Picture down	The **picture down** key scrolls the document window one screen page downwards, while the cursor remains in the same position in the document window.
Ctrl+Picture up	When the **Ctrl** key is pressed, the **Picture up** key scrolls the document window one page to the left while the cursor remains in the same place in the document window.
Ctrl+Picture down	When the **Ctrl** key is pressed, the **Picture down** key scrolls the document window one page to the right while the cursor remains in the same place in the document window.

Edit

Edit

Key	Function
Space bar	In select mode, the object at the cursor position is selected and all other objects are deselected. In placing mode the corresponding object is placed where the cursor is.
Shift key+**Space bar**	In selection mode, when the **Shift** key is pressed, objects which have not previously been selected in the cursor position are selected, or vice versa. The selection of all other objects is not affected. In placing mode the corresponding object is placed where the cursor is.
Space bar+**Cursor Keys**	In selection mode – if there is no selected object where the cursor is – the cursor moves and a selection rectangle is displayed. If a selected object is in the cursor position, all objects will be shifted according to how the cursor is moved. The number of inputs of an FFB with a variable input number can be changed in the FB Editor’s Selection Mode by placing the cursor on the rectangle in the middle of the lower edge of the selection frame, which holds down the **Space bar** and presses the **Up** or **Down** keys. The width of the branches or connections can be changed in the SFC Editor’s Selection Mode by placing the cursor on the rectangle of the selection frame, which holds down the **Space bar** and presses the **Right** or **Left** keys. In Link Mode, a link is produced by dragging the mouse.
Shift key+**Space bar**+**Cursor keys**	In Selection Mode, this key combination creates a selection frame as described above, and the selection of all other objects is retained.

Allocating variables onto an FFB

To allocate variables onto an FFB, do the following:

Step	Action
1	Use the **cursor keys** or **Shift**+**cursor keys** to move the cursor to the input/output of the FFB.
2	Press **Enter**. **Reaction:** The **Connect FFB** dialog for the selected input/output opens.

Changing variables onto an FFB

To change variables onto an FFB, do the following:

Step	Action
1	Use the **cursor keys** or **Shift**+**cursor keys** to move the cursor to the FFB variables to be changed.
2	Press **Enter**. **Reaction:** The **Connect FFB** dialog for the selected input/output opens.

Changing the number of inputs/outputs

To change the number of inputs/outputs with extendable FFBs, do the following:

Step	Action
1	Use the **cursor keys** or **Shift**+**cursor keys** to move the cursor to the centre of the lower edge of the FFB's block frame.
2	Press **Space bar**+**Down cursor key** to generate further inputs/outputs. Press **Space bar**+**Up cursor key** to hide further inputs/outputs. **Reaction:** The number of inputs/outputs is changed.

Shortcut keys in the LD-Editor

At a Glance

Concept supports the work with the keyboard in the graphic editors. Although the mouse is a more appropriate input tool, it is nevertheless possible to operate Concept with the keyboard alone – especially in machine environments. The Editors behave in the same way regardless of whether they are operated with the mouse or with the keyboard.

Rules

The following general rules need to be observed:

- The space bar corresponds to the left mouse button, i.e. the space bar is used for selecting and moving.
- The Enter key corresponds to the double click with the left mouse button – for example, the input key is used to call up the properties dialog of objects.
- The Shift key is used in conjunction with the keyboard exactly as it is with the mouse – for example, the Shift key is used to extend an object selection or to reselect a few objects from a number which have already been selected.
- Pressing a key only once only affects the element in the center of the current cell.
- Pressing a key together with **Ctrl** affects the right side of the current cell..
- Striking a key together with **Shift** afects the left side of the current cell

Calling up menu command

The table shows the additional shortcut keys and their corresponding menu commands avialable in LD Editor (see also *General Short Cut Keys, page 833*):

Key	**Menu Entry Command Executed**
Ctrl+A	**Edit →Select All**
Ctrl+Z	**Edit →Undo delete**
Ctrl+X	**Edit →Cut**
Ctrl+C	**Edit →Copy**
Ctrl+V	**Edit →Paste**
Del	**Edit →Delete**
Ctrl+O	**View →Overview**
Ctrl+N	**View →Normal**
Ctrl++	**View →Zoom in**
Ctrl+-	**View →Zoom out**
Esc	**Objects →Select**
Shift+H	**Objekts →Link**
H	**Objects →Direct Link**

Key	Menu Entry Command Executed
V	**Objects →Vertical Link**
F	**Objects →FFB: Last Type**
I	**Objects →Invert Input/Output**
T	**Objects →Text**
Shift+F	**Objects →FFB selection...**
C	**Objects →Contact Normally Open**
L	**Objects →Contact – Normally Closed**
P	**Objects →Contact - Rising Edge (Positive)**
N	**Objects →Contact - Falling Edge (Negative)**
Shift+C	**Objects →Coil**
Shift+L	**Objects →Coil - Negated**
Shift+S	**Objects →Coil - Set**
Shift+R	**Objects →Coil - Reset**
Shift+P	**Objects →Coil - Rising Edge (Positive)**
Shift+N	**Objects →Coil - Falling Edge (Negative)**
Ctrl+Y	**Online →Animate booleans**
Ctrl+W	**Online →Animate selected**

Placing objects

In order to place objects in the LD Editor by using the keyboard, please carry out the following steps:

Step	Action
1	Move the field with a gray background onto the field where the object is to be placed (move gray field (selecting a field)).
2	Strike the key assigned to the object (see *Creating objects, page 845*). **Reaction:** Adjoining boolean objects are automatically connected.
3	Links between non-adjoining objects and non-boolean in/outputs have to be made with the mouse pointer (see *Moving the mouse pointer, page 845*).
4	The mouse pointer must also be used to invert in/outputs (see *Moving the mouse pointer, page 845*).

Moving the gray field (selecting a field)

Moving the gray field (selecting a field)

Key	Function
Up	Moves the gray field up by one field
Down	Moves the gray field down by one field
To the right	Movesthe gray fields to the right by one field
To the left	Moves the gray fields to the left by one field
Home	Moves the gray field to the left margin
Shift+Home	Moves the gray field to the left margin
End	Moves the gray field to the right margin
Shift+End	Moves the gray field to the right margin
Ctrl+Home	Moves the gray field to the top left-hand corner
Ctrl+End	Moves the gray field to the top right-hand corner

Selecting objects

Selecting objects

Key	Function
Space character	Selects object in the middle of the gray field
Ctrl+Space character	Selects object on the right-hand side of the gray field
Shift+Space character	Selects object on the left-hand side of the gray field
Enter	In select mode: Selects object in the middle of the gray field and opens its Select dialog (if available)
Ctrl+Enter	In select mode: Selects object from the right-hand side of the gray field and opens its Select dialog (if available)
Shift+Enter	In select mode: Selects object from the left-hand side of the gray field and opens its Select dialog (if available)

Moving a selected object

Moving a selected object

Key	Function
Shift+**Up**	Moves the selected object up by one field
Shift+**Down**	Moves the selected object down by one field
Shift+**Right**	Moves the selected object to the right by one field
Shift+**Left**	Moves the selected object to the left by one field

Allocating variables onto an FFB

To allocate variables onto an FFB, do the following:

Step	Action
1	Move the gray field onto the cell containing the in/output.
2	To allocate variables to inputs, press **Ctrl**+**Enter**. To allocate variables to outputs press **Ctrl**+**Enter**. **Reaction:** The dialog **Connect FFB** of the selected in/output is opened.

Changing variables onto an FFB

To change variables onto an FFB, do the following:

Step	Action
1	Move the gray field onto the cell containing the variable to be changed.
2	To select the variable press **Shift**+**Enter**. **Reaction:** The dialog **Connect FFB** of the selected in/output is opened.

Deleting vertical links

To delete vertical variables, carry out the following step:

Step	Action
1	Move the gray field onto the cell running through the vertical link.
2	Press **Ctrl**+**Delete**. **Reaction:** The vertical link is deleted.

Moving the mouse pointer

Moving the mouse pointer

Key	Function
Ctrl+Up	Moving the mouse pointer up by one step
Ctrl+Down	Moving the mouse pointer down by one step
Ctrl+Right	Moving the mouse pointer to the right by one step
Ctrl+Left	Moving the mouse pointer to the left by one step

Scrolling

Scrolling:

Key	Function
Picture up	Scrolls the display sector one page up
Shift+Picture up	Scrolls the display sector one page up
Picture down	Scrolls the display sector one page down
Shift+Picture down	Scrolls the display sector one page down
Ctrl+Picture up	Scrolls the display sector one page to the right
Ctrl+Picture down	Scrolls the display sector one page to the right

Creating objects

Creating objects

Key	Function
C	Creates a N.O. in the gray field
L	Creates an opener in the gray field
P	Creates a contract for the recognition of positive flanks in the gray field
N	Creates a contract for the recognition of negative flanks in the gray field
Shift+C	Creates a coil in the gray field
Shift+L	Creates a negated coil in the gray field
Shift+S	Creates a coil set in the gray field
Shift+R	Creates a reset coil in the gray field

Key	Function
Shift+P	Creates a coil for the recognition of positive flanks in the gray field
Shift+N	Creates a coil for the recognition of negative flanks in the gray field
Shift+F	Opens FFB selection dialog
F	Creates current FFB in the gray field

Creating links

Creating links

Key	Function
H	Activates the link mode
V	Creates a vertical link in the right-hand bottom corner of the gray field (and then moves the gray field to the right by one field)
Shift+V	Creates a vertical link in the bottom left-hand corner of the gray field.

Activating the different modes

Activating the different modes

Key	Function
Space character	Activates the selection mode
Esc	Activates the selection mode
H	Activates the link mode
I	Activates the mode for inverting in/outputs
T	Activates the text mode

Short Cut Keys in the LL984-Editor

Short Cut Keys

The table shows the additional short cut keys available in the LL984 editor and the corresponding menu entry commands (see also *General Short Cut Keys, page 833*):

Short Cut Keys	Menu Entry Command Executed
Ctrl+Z	Edit →Undo delete
Ctrl+X	Edit →Cut
Ctrl+C	Edit →Copy
Ctrl+V	Edit →Paste
Del	Edit →Delete
Ctrl+D	Edit →DX Zoom...
Ctrl+H	Edit →Offset References...
Ctrl+O	View →Overview
Ctrl+N	View →Normal
Ctrl+E	View →Expanded
Ctrl++	View →Zoom in
Ctrl+-	View →Zoom out
(	Objects →Coil
Ctrl+L	Objects →Coil - Retentive
"	Objects →Contact – Normally Open
/	Objects →Contact – Normally Closed
P	Objects →Contact – Pos Trans
N	Objects →Contact – Neg Trans
=	Objects →Horiz Short
I	Objects →Vertical Short
Ctrl+F	Objects →Instruction by name...
Ctrl+G	Network →Goto...
Ctrl+I	Networks →Insert
Ctrl+Q	Networks →Insert Equation
Ctrl+A	Networks →Append
Ctrl+U	Networks →Attach formula
Ctrl+K	Networks →Delete
Picture up	Networks →Next

Short Cut Keys	Menu Entry Command Executed
Picture up	Networks →Previous
Ctrl+M	Networks →Comment...
Ctrl+T	Online →Trace
Ctrl+B	Online →ReTrace

IEC conformity

D

Overview

This Chapter contains the standards tables required by IEC 1131-1.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
D.1	What is the IEC 1131-3 standard?	850
D.2	IEC standards tables	853
D.3	Expansions of IEC 1131-3	873
D.4	Text language syntax	875

D.1 What is the IEC 1131-3 standard?

Overview

This section contains general information about IEC 1131-3 and the implemented IEC conformity test.

What's in this Section?

This section contains the following topics:

Topic	Page
General information about IEC conformity	851
IEC Conformity Test	852

General information about IEC conformity

At a Glance

The IEC standard 1131-3 (compare chapter 1.4) specifies the syntax and semantics of a standardized series of programming languages for Programmable Logic Controls (PLC). These include the two text languages IL (Instruction List) and ST (Structured Text) and the two graphical languages LD (Ladder Diagram) and FBD (Function Block Diagram).

It also defines the elements of the sequential function chart (SFC) language for structuring the internal organization of PLC programs and Function Blocks. Configuration elements, used for installing PLC programs onto PLC systems, are also defined.

NOTE: Concept uses the English acronyms for the programming languages.

Furthermore, it defines methods to enable communication between the PLC and other automated system components.

Concept standard accordance

In accordance with the standard, the present version of the programming system Concept supports a subset of language elements, which are defined in the standard.

In this context, accordance with the standard means the following:

- The standard allows the individual implementing an IEC program system to select or deselect certain language properties or even complete languages from the selection tables, which represent an integrated part of the standard specifications. A system, which itself accords with the standard, may only implement the selected properties exactly as they are given in the standard.
- In addition, the standard enables the individual implementing to introduce defined language elements into an interactive programming environment. As the standard expressly emphasizes that the specification of such environments lies outside of its area of application, the person implementing has a certain degree of freedom to offer optimized forms of display and implementation mechanisms for the benefit of the user.
- Concept uses these degrees of freedom e.g. when introducing the term "Project" to implement the IEC language elements "Configuration", "Resource" and "Program" all together (Concept only supports one single cyclically running program within a single resource within the configuration). Apart from this, it uses them, for example, with implementation mechanisms made available for declaring variables and authorizing Function Blocks.

IEC standards tables

Information on which properties are supported and other implementation specific details can be found in the following statements on standard fulfilment and the associated standards tables.

IEC Conformity Test

Testing the Import/Export Interface

An interface for importing standard IEC programs and DFBs from ASCII files (menu **File →Import**) and exporting these programs into graphical languages in ASCII format (menu **File →Export**) is available in Concept. The conformity of this interface can be tested using files which can be obtained from IFAK (Institut für Automation und Kommunikation e.V. Magdeburg).

IEC conformity test scripts:

(c) 1994, IFAK Institut für Automation und Kommunikation e.V.

Magdeburg

Steinfeldstraße 3

D-39179 Barleben

Notes

The following points must be considered with regard to the conformity of the import interface:

- In Concept, IL operators are permitted as identifiers.
  R, S, LD, S1 and R1 are possible parameter names. Therefore, there will be no changes made to the standard functions/function blocks. Concept requires no change in the IEC table 54 with S to SET, R to RESET, S1 to SET1, R1 to RESET1.
- All IL operators not in conflict with functions are permitted as variable names in Concept (N, S, R, S1, R1, CLK, CU, CD, PV, IN, PT) – contrary to IEC table 54.
- Counter EFBs must be typified in Concept, e.g. CTU must become CTU_INT.
- Function block instances cannot be called up more than once; a restriction that is self-evident if IEC table 53, property 3 is required.
- An overflow of time span variables (e.g. t#100s) is not detected. The system calculates the time correctly, so that detection of an overflow is not necessary.
- IEC IL comments are only permitted as the last element in a line. Concept allows comments to be made everywhere.

D.2 IEC standards tables

Overview

This system fulfils the requirements of the IEC 1131-3 in the following properties of the language.

What's in this Section?

This section contains the following topics:

Topic	Page
Common elements	854
IL (AWL) language elements	861
ST language elements	863
Common graphic elements	865
LD (KOP) language elements	866
Implementation-dependent parameters	868
Error causes	871

Common elements

IEC standards table

IEC standards table for common elements:

Table number	Property number	Property description
1	1	For required character set – see *Chapter 2.1.1* of 1131-3
1	2	Lower case characters
1	3a	Hash key (#)
1	4a	Dollar sign ($)
1	5a	Vertical line (|)
1	6a	Left and right square brackets "[ ]"
2	1	Upper case character and numbers
2	2	Upper and lower case characters, numbers, embedded underscore
2	3	Upper and lower case characters, numbers, leading and embedded underscore
3	1	Comments
4	1	Integer (whole number) literals
4	2	Real literals
4	3	Real literals with exponents
4	4	Base 2 literals
4	5	Base 8 literals
4	6	Base 16 literals
4	7	Boolean zero and one
4	8	Boolean FALSE and TRUE
7	1a	Time span without underscores: short prefix
7	1b	Time span without underscores: long prefix
7	2a	Time span with underscores short prefix
7	2b	Time span with underscores long prefix
10	1	BOOL: Boolean
10	3	INT: Integer
10	4	DINT: Double integer
10	7	UINT: Signed integer
10	8	UDINT: Signed double integer
10	10	REAL: Floating point number

Table number	Property number	Property description
10	12	TIME: Time span
10	17	BYTE: Bit sequence 8
10	18	WORD: Bit sequence 16
12	4	Data types for fields
12	5	Data types for structures
15	1	I: Input (*Note 1, page 859*)
15	2	Q: Output (*Note 2, page 859*)
15	4	X: Bit size (*Note 2, page 859*, *Note 1, page 859*)
15	5	no prefix: Bit size (*Note 2, page 859*, *Note 1, page 859*)
15	6	B: Byte size (*Note 2, page 859*, *Note 1, page 859*)
15	7	W: Word size (*Note 2, page 859*, *Note 1, page 859*)
15	8	D: Double word size (*Note 2, page 859*, *Note 1, page 859*)
17	2	Declaration of directly displayed buffered variables (*Note 5, page 860*, *Note 9, page 860*)
17	3	Declaration of storage locations with symbolic variables (*Note 5, page 860*)
17	4	Assignment of storage locations with fields (*Note 5, page 860*, *Note 11, page 860*)
17	5	Automatic storage allocation for symbolic variables (*Note 5, page 860*)
17	7	Declaration for buffered fields (*Note 5, page 860*, *Note 11, page 860*)
17	8	Declaration for structured variables (*Note 5, page 860*)
18	2	Initialization of directly displayed buffered variables (*Note 5, page 860*, *Note 9, page 860*, *Note 10, page 860*)
18	3	Assignment of storage locations and start values for fields(*Note 5, page 860*)
18	4	Assignment of storage locations and start values for fields (*Note 5, page 860*, *Note 11, page 860*)
18	5	Initialization of symbolic variables (*Note 5, page 860*)
18	7	Declaration and initialization of buffered variables (*Note 5, page 860*, *Note 11, page 860*)
18	8	Initialization of structured variables (*Note 5, page 860*)
18	9	Initialization of constants
19	1	Negated input
19	2	Negated output

Table number	Property number	Property description
20	1	Use of "EN" and "ENO" - REQUIRED for LD (*Note 6, page 860*)
20	2	Use of "EN" and "ENO" – OPTIONAL for FBD
20	3	FBD without "EN" and "ENO"
21	2	Standardized functions (*Note 3, page 859*)
22	1	(*-TO-**) Type conversion functions (*Note 4, page 859*
22	2	Truncation towards zero: TRUNC (*Note 3, page 859*))
23	1	ABS: Absolute value
23	2	SQRT: Square root
23	3	LN: Natural logarithm
23	4	LOG: Base 10 logarithm
23	5	EXP: Exponential function
23	6	SIN: Sine, input in radians
23	7	COS: Cosine, input in radians
23	8	TAN: Tangent, input in radians
23	9	ASIN: Arc sine, principal value
23	10	ACOS: Arc cosine, principal value
23	11	ATAN: Arc tangent, principal value
24	12	ADD: Add
24	13	MUL: Multiply
24	14	SUB: Subtract
24	15	DIV: Divide
24	16	MOD: Modulo
24	17	EXPT: Exponentiation
24	18	MOVE: Assignment
25	1	SHL: move to the left
25	2	SHR: Move to the right
25	3	ROR: Rotate to the right
25	4	ROL: Rotate to the left
26	5	AND: LLogical And
26	6	OR: Logical Or
26	7	XOR Logical exclusive Or
26	8	NOT: Negation
27	1	SEL: Binary selection

Table number	Property number	Property description
27	2a	MAX: Extendable maximum
27	2b	MIN: Extendable minimum
27	3	LIMIT: Limit
27	4	MUX: Extendable multiplexer
28	5	GT: Falling sequence
28	6	GE: Monotonic sequence (decreasing)
28	7	EQ: Equality
28	8	LE: Monotonic sequence (increasing)
28	9	LT: Rising seqence
28	10	NE: Inequality
30	1	ADD: Adding TIME to TIME
30	4	SUB: Subtracting TIME from TIME
30	10	MUL: Multiplying TIME by ANY_NUM
30	11	DIV: Dividing TIME by ANY_NUM
33	1	RETAIN identifier for internal variables (*Note 5, page 860*)
33	2	RETAIN identifier for output variables (*Note 5, page 860*)
33	3	RETAIN identifier for internal Function Blocks (*Note 5, page 860*)
34	1	Bistable Function Block (set priority)
34	2	Bistable Function Block (reset priority)
35	1	Detecting the rising edge
35	2	Detecting the falling edge
36	1	Up counter
36	2	Down counter
36	3	Up/Down counter
37	1	TP: Pulse (timer)
37	2a	TON: Switch-on delay
37	3a	TOF: Switch-off delay
39	1	RETAIN identifier for internal variables (*Note 5, page 860*)
39	2	RETAIN identifier for output variables (*Note 5, page 860*)
39	3	RETAIN identifier for internal Function Blocks (*Note 5, page 860*)
39	14	Assignment of storage locations with fields (*Note 5, page 860*)

Table number	Property number	Property description
39	18	Assignment of storage locations and start values for fields(*Note 5, page 860*)
39	19	Use of directly displayed variables (*Note 2, page 859*, *Note 1, page 859*)
40	1	Step/Start step – graphical form with directional links
40	2	Step/Start step – text form without directional links (Note 8)
40	3a	Step marker – general form
40	4	Step time elapsed – general form
41	1	Transition condition in ST language within the graphic (*Note 8, page 860*)
41	5	Transition condition in ST language – textual reference (*Note 9, page 860*)
41	6	Transition condition in IL language – textual reference (*Note 9, page 860*)
41	7	Use of the transition name
41	7b	Transition condition in FBD language
41	7c	Transition condition in IL language
41	7d	Transition condition in ST language
42	1	Each Boolean variable can be an action
43	1	Action block
43	2	Concatenated action blocks
43	3	Step body in text form (*Note 8, page 860*)
44	1	Identifier
44	2	Action name
45	1	Not saved (no identifier)
45	2	N: not saved
45	3	R: Overriding reset
45	4	S: Set (saved)
45	5	L: Time limited
45	6	D: Delayed
45	7	P: Pulse
45	9	DS: Delayed and saved
46	1	Simple string
46	2a	Branching in string selection (priority from left to right)
46	3	Merging a string selection

Table number	Property number	Property description
46	4	Parallel strings - branch and merge
46	5a	String jump (priority from left to right)
46	6a	String loop (priority from left to right)

Note 1

Modicon TSX Quantum Präfix 3 is used in the prefix IB, ID position in all graphical languages.

Note 2

Modicon TSX Quantum Präfix 4 is used in the prefix QB, QD position in all graphical languages.

Note 3

The following functions are overloaded with reference to the data which is selected, multiplexed or assigned; the type statement refers to the selection parameters.

List of overloaded functions:

- SEL
- MUX
- MOVE

All other functions are standardized, e.g. REAL_TRUNC_INT.

Note 4

List of type conversion functions:

- BOOL_TO_BYTE, BOOL_TO_DINT, BOOL_TO_INT, BOOL_TO_REAL, BOOL_TO_TIME, BOOL_TO_UDINT, BOOL_TO_UINT, BOOL_TO_WORD,
- BYTE_TO_BOOL, BYTE_TO_DINT, BYTE_TO_INT, BYTE_TO_REAL, BYTE_TO_TIME, BYTE_TO_UDINT, BYTE_TO_UINT, BYTE_TO_WORD,
- DINT_TO_BOOL, DINT_TO_BYTE, DINT_TO_INT, DINT_TO_REAL, DINT_TO_TIME, DINT_TO_UDINT, DINT_TO_UINT, DINT_TO_WORD,
- INT_TO_BOOL, INT_TO_BYTE, INT_TO_DINT, INT_TO_REAL, INT_TO_TIME, INT_TO_UDINT, INT_TO_UINT, INT_TO_WORD,
- REAL_TO_BOOL, REAL_TO_BYTE, REAL_TO_DINT, REAL_TO_INT, REAL_TO_TIME, REAL_TO_UDINT, REAL_TO_UINT, REAL_TO_WORD,
- TIME_TO_BOOL, TIME_TO_BYTE, TIME_TO_DINT, TIME_TO_INT, TIME_TO_REAL, TIME_TO_UDINT, TIME_TO_UINT, TIME_TO_WORD,
- UDINT_TO_BOOL, UDINT_TO_BYTE, UDINT_TO_DINT, UDINT_TO_INT, UDINT_TO_REAL, UDINT_TO_TIME, UDINT_TO_UINT, UDINT_TO_WORD,

- UINT_TO_BOOL, UINT_TO_BYTE, UINT_TO_DINT, UINT_TO_INT, UINT_TO_REAL, UINT_TO_TIME, UINT_TO_UDINT, UINT_TO_WORD,
- WORD_TO_BOOL, WORD_TO_BYTE, WORD_TO_DINT, WORD_TO_INT, WORD_TO_REAL, WORD_TO_TIME, WORD_TO_UDINT, WORD_TO_UINT

The consequences of each conversion are described in the block library and the help texts, which are available for the library of IEC standard functions.

Note 5

The RETAIN identifier is implicitly required; no language elements displayed in non-buffered memory areas are supported.

Note 6

"EN" and "ENO" are offered as standard; they can, however, be hidden and any other input or output of data type BOOL can be used for links.

Note 7

Expressions are restricted to individual simple Boolean variables.

Note 8

Only available on import of IEC text form in graphical SFC representation.

Note 9

Only available in textual declaration in IL or ST sections.

Note 10

Initialization only possible for non Boolean outputs.

Note 11

Declaration of field variables only possible when using previously defined field data type names.

IL (AWL) language elements

IEC standards table

IEC standards table for IL (AWL) language elements:

Table number	Property number	Property description
52	1	LD operator: sets the current result to that of the operand
52	2	ST operator: saves the current result to the operand address
52	3	S operator: sets Boolean operands to "1" R operator: sets Boolean operands to "0"
52	4	AND operator
52	6	OR operator
52	7	XOR operator
52	8	ADD operator
52	9	SUB operator
52	10	MUL operator
52	11	DIV operator
52	12	GT operator: Comparison >
52	13	GE operator: Comparison >=
52	14	EQ operator: Comparison =
52	15	NE operator: Comparison <>
52	16	LE operator: Comparison <=
52	17	LT operator: Comparison <
52	18	JMP operator: Jump to tag (*Note 1, page 861*)
52	19	CAL operator: Calls Function Block
52	21	Closing bracket ")": Editing deferred operations
53	1	CAL operator with list of input parameters
53	2	CAL operator with loading/saving of input parameters

Note 1

Jumps are only allowed within sections, not across section boundaries.

Note 2

The following keywords are not available:

- TYPE...END_TYP
- VAR_INPUT...END_VAR
- VAR_OUTPUT...END_VAR
- VAR_IN_OUT...END_VAR
- VAR_EXTERNAL...END_VAR
- FUNCTION...END_FUNCTION
- FUNCTION_BLOCK...END_FUNCTION_BLOCK
- PROGRAM...END_PROGRAM
- STEP...END_STEP
- TRANSITION...END_TRANSITION
- ACTION...END_ACTION
- SEGMENT_SCHEDULER
- RET
- &

ST language elements

IEC standards table

IEC standards table for ST language elements:

Table number	Property number	Property description
55	1	Placing in brackets: (Expression)
55	2	Function calls: Function name (list of arguments)
55	3	Exponentiation: **
55	4	Negation: -
55	5	Complement: NOT
55	6	Multiplication: *
55	7	Division: /
55	8	Modulo: MOD
55	9	Addition: +
55	10	Subtraction: -
55	11	Comparison: <, >, <=, >=
55	12	Equality: =
55	13	Inequality: <>
55	14	Boolean AND: &
55	15	Boolean AND: AND
55	16	Boolean exclusive OR: XOR
55	17	Boolean OR: OR
56	1	Assignment
56	2	Function Block calls and use of FB outputs
56	4	IF instruction
56	5	CASE instruction
56	6	FOR instruction
56	7	WHILE instruction
56	8	REPEAT instruction
56	9	EXIT instruction
56	10	Empty instruction

Note 1

The following keywords are not available:

- TYPE...END_TYP
- VAR_INPUT...END_VAR
- VAR_OUTPUT...END_VAR
- VAR_IN_OUT...END_VAR
- VAR_EXTERNAL...END_VAR
- FUNCTION...END_FUNCTION
- FUNCTION_BLOCK...END_FUNCTION_BLOCK
- PROGRAM...END_PROGRAM
- STEP...END_STEP
- TRANSITION...END_TRANSITION
- ACTION...END_ACTION
- SEGMENT_SCHEDULER
- RETURN

Common graphic elements

IEC standards table

IEC standards table for common graphic elements:

Table number	Property number	Property description
57	2	Horizontal lines: Graphic or semi-graphic
57	4	Vertical lines: Graphic or semi-graphic
57	6	Horizontal/vertical connection: Graphic or semi-graphic
57	8	Line intersection without connection: Graphic or semi-graphic
57	10	Connected and unconnected corners: Graphic or semi-graphic
57	12	Blocks with connecting lines: Graphic or semi-graphic

LD (KOP) language elements

IEC standards table

IEC standards table for LD (KOP) language elements:

Table number	Property number	Property description
59	1	Left power rail (with linked horizontal connection)
60	1	Horizontal connection
60	2	Vertical connection (with linked horizontal connections)
61	1	Closer
61	3	Opener
61	5	Contact for detection of positive transition
61	7	Contact for detection of negative transition
62	1	Coil (*Note 1, page 866*)
62	2	Negative coil (*Note 1, page 866*)-{}-
62	3	SET coil (*Note 1, page 866*)
62	4	RESET coil (*Note 1, page 866*)
62	8	Coil for detection of positive transition
62	9	Coil for detection of negative transition

Note 1

In start behavior of PLCs there is a distinction between cold starts and warm starts:

- **Cold start**
  Following a cold start (loading the program with **Online** →**Load**) all variables (irrespective of type) are set to "0" or, if available, their initial value.
- **Warm start**
  In a warm start (stopping and starting the program or **Online** →**Load changes**) different start behaviors are valid for located variables/direct addresses and unlocated variables:
  - **Located variables/direct addresses**
    In a warm start all 0x, 1x and 3x registers are set to "0" or, if available, their initial value.
    4x registers retain their current value (storage behavior).
  - **Unlocated variables**
    In a warm start all unlocated variables retain their current value (storing behavior).

This varying behavior in a warm start leads to peculiarities in the warm start behavior of set and reset functions.

- **Set and Reset in LD and IL**
  Warm start behavior is dependent on the variable type used (storage behavior in use of unlocated variables; non storage behavior in use of located variables/direct addresses)
- **SR and RS Function Blocks in FBD, LD, IL and ST**
  These Function Blocks work with internal unlocated variables and therefore always have a storage behavior.

Implementation-dependent parameters

IEC standards table

IEC standards table for implementation-dependent parameters:

Parameters	**Threshold values/behavior**
Error-handling procedure	See *Error causes, page 871* & EFB help
National characters used	All characters in the Windows ANSI character set are supported.
Maximum length of identifiers	Program name: 8 Formal parameter names: 8 DFB type names: 8 EFB type names: 17 Data type names: 24 all others: 32
Maximum comment length:	Limited only by Windows resources
Range of values for time span literals	0s to 49d_17h_2m_47.295s
Range of values for variables of type TIME	0s to 49d_17h_2m_47.295s
Accuracy of the seconds display with types TIME_OF_DAY and DATE_AND_TIME	not applicable
Maximum number of field indices	Practically no limit
Maximum field size	64 kB
Maximum number of structure elements	Only limited by Windows or PLC resources
Maximum structure size	64 kB
Maximum number of variables per declaration	Only limited by Windows or PLC resources
Maximum number of enumerated values	not applicable
Default maximum length of STRING variables	not applicable
Maximum authorized length of STRING variables	not applicable
Maximum number of hierarchy tiers	1
Configured or physical illustration	Configured illustration, physical illustration through separate I/O projection
Parameters	Threshold values/behavior
Maximum number of indices	Practically no limit
Maximum range of index values	Range of data type INT

Parameters	**Threshold values/behavior**
Maximum number of structure levels	Only limited by Windows or PLC resources
Initialization of system inputs	System zero; no user-definable start values
Maximum number of variables per declaration	Only limited by Windows or PLC resources
Information for the determination of execution times of program organization units	In preparation
Methods of function display (names or symbols)	Names
Maximum number of function specifications	not applicable
Maximum number of inputs for extendable functions	32
Type conversion accuracy	See EFB help
Accuracy of functions of a variable	INTEL floating point processor or emulator
Arithmetic function implementation	INTEL floating point processor or emulator
Maximum number of Function Block specifications	Only limited by Windows or PLC resources
Maximum number of Function Block authorizations	512 per section; number of sections per program organization unit is only limited by Windows or PLC resources
Pvmin, Pvmax of counters	Limited by rangess of the INT or DINT data types
Effect of a change in the value of a PT input during a time measurement operation	Directly affects the timer’s default time
Program size limits	Only limited by available PLC memory
Time behavior and porting effects of the execution control elements	The execution of SFC networks in different sections occurs sequentially, in the order given in these sections.
Accuracy of elapsed step time	10 ms
Maximum number of steps per SFC	Limited by the available area for entering characters within the section; number of sections per program organization unit only limited by Windows or PLC resources; the upper limit for the total number of objects per SFC is 2000
Parameters	Threshold values/behavior

Parameters	**Threshold values/behavior**
Maximum number of transitions per SFC and per step	Limited by the available area for entering characters within the section; number of sections per program organization unit only limited by Windows or PLC resources; the upper limit for the total number of objects per SFC is 2000
Action control mechanism	Functionally equivalent to the specification in the standard
Maximum number of actions per step	Only limited by Windows or PLC resources

Graphical display of the step situation	Green = active
	Red = inactive

Transition switch time	Of the magnitude of 10 ms
Maximum width of branches/connections	Limited by the available area for entering characters 32
Contents of the RESOURCE libraries	See EFB libraries & help
Maximum number of tasks	1
Task interval resolution	not applicable
Pre-justified and non pre-justified schedules	not applicable
Maximum length of expressions	Practically no limit
Partial evaluation of Boolean expressions	no partial evaluation
Maximum length of instructions	Practically no limit
Maximum number of CASE selections	Practically no limit
Value of the control variables on completion of FOR loops	undefined
Graphic/semi-graphic display	Graphic
Network topology restrictions	no restrictions
Evaluation sequence of feedback loops	Within a network, the starting point of the FFB execution sequence is determined by the "single" available feedback variable
Means of specifying the network execution sequence	1: Execution sequence of program organization unit sections 2: The network execution sequence can be changed within sections; this is done by using a menu command to switch between the execution sequences of two selected FFB items

Error causes

IEC standards table

IEC standards table for error causes:

Error cause	**Handling (see *Note 1, page 872*)**
Variable value exceeds the specified range	not applicable
Initialization list length and number of field elements do not agree	2) Error message during programming
Incorrect use of directly displayed or external variables in functions	not applicable
Type conversion error	4) Error message during execution
Numerical result exceeds the range for data type	4) Error message during execution
Division by zero	4) Error message during execution
Mixed input data types in a selection function	2) Error message during programming
Selector (K) outside MUX function range	4) Error message during execution
Invalid character position	not applicable
Result exceeds maximum sequence length	not applicable
Numerical result exceeds the range for data type	4) Error message during execution
Zero or more than one starting step in SFC network	3) Error message during analysis/loading/connection
User program attempting to change step situation or step time	2) Error message during programming
Simultaneously completed transitions without priority in a selection branch	not applicable
Side effects of evaluation of a transition condition	3) Error message during analysis/loading/connection
Action control error	1) Error not reported
Unsafe or unreachable SFCs	3) Error message during analysis/loading/connection
Data type conflict in VAR_ACCESS	not applicable
Tasks demanding too many processor resources	3) Error message during analysis/loading/connect
Scan time overrun	4) Error message during execution
Error cause	Handling (see note 1)
Further task schedule conflicts	not applicable

Error cause	Handling (see *Note 1, page 872*)
Numerical result exceeds the range for data type	4) Error message during execution
Division by zero	4) Error message during execution
Invalid data type for operation	3) Error message during analysis/loading/binding
Return from function without assigned value	not applicable
Occurrence arrives at no outcome	4) Error message during execution
The same identifier as connector tag and element name use	not applicable
Non-initialized feedback variable (initialized with system zero)	1) Error not reported

Note 1

Identification for the handling of error causes according to IEC 1131-3, chapter 1.5.1, d):

- 1) Error not reported
- 2) Error message during programming
- 3) Error message during analysis/loading/binding
- 4) Error message during execution

D.3 Expansions of IEC 1131-3

Expansions of IEC 1131-3

At a Glance

The Concept programming environment makes the construct of the so-called section available in all programming languages permitting the subdivision of a program organization unit. This construct provides the opportunity to mix several languages in the body of a POU (e.g. FBD sections, SFC sections), a property, which, if used for this purpose, represents an expansion of the IEC syntax. Sections do not generate their own name space; the name space for all language elements is the POU.

Sections appearing in the body of a POU written only in the FBD language are not to be viewed as an expansion, rather as a permitted means of specifying the execution sequences of several FBD networks furnished with tags, as specified in the corrigendum to 1131-3.

Purpose of sections

Sections serve various purposes

- Sections permit the functional division of an expansive POU body: The body of a POU can be divided into sensible functional parts. The section list represents a kind of functional table of contents for a large, otherwise unstructured POU body.
- Sections permit the graphical division of an expansive POU body: in accordance with an intentionally graphic form of representation, sub-structures of an expansive body can be established. Smaller or larger partial structures may be chosen.
- The division of an expansive POU body enables faster online changes: the section serves as the unit for online changes in Concept. If the POU body is changed in various places during the program runtime, all sections affected by the changes are taken into account if explicitly initiated reloading occurs.

- Sections permit the execution sequence to influence particular marked parts of the POU body: the section name serves as a marking for the part of the body contained in the section, and the execution sequence of the sections can be changed by ranking the sections (see also the last part of the "implementation-dependent parameters" table for information on the execution sequence of networks in the FBD language).
- Sections permit the parallel use of different languages in the same POU: this property is a considerable expansion of the syntax of the IEC 1131-3 standard, which only permits the use of a single IEC language for a POU body. Only the SFC language also provides the opportunity to formulate parts of the body in different languages, because transitions and actions can be expressed in any language, in as far as the corresponding properties are supported by the programming system.

D.4 Text language syntax

Text Language Syntax

Description

The programming system Concept supports the complete language syntax, as specified in appendix B of the IEC language standard 1131-3, with the following exceptions:

- Syntax productions in appendix B of 1131-3, belonging to properties, which according to the IEC standards tables in *IEC standards tables, page 853* in this document are not supported by Concept, are not implemented.
- The use of some Concept supported properties is, according to the associated remarks in the IEC standards table, only possible in a restricted or modified form. The associated syntax productions are therefore only occasionally or somewhat differently implemented.
- Concept supports the NOT Operator for inverting Boolean battery content in IL.
- The implementation of some faulty syntax productions in appendix B of 1131-3, improved upon either in the corrigendum to 1131-3 or in the planned amendment to 1131-3, uses the suggestions in these documents for orientation.
  The improved productions are implemented in Concept as follows (chapter numbers refer to appendix B of 1131-3):
  - **B.1.3.3:**
    ```
    array_initialization ::= ’[’ array_initial_elements {’,’
    array_initial_elements} ’]’

    initialized_structure ::= structure_type_name [’:=’
    structure_initialization]
    ```
  - **B.2.1:**
    ```
    il_operand_list ::= il_operand [’,’ [EOL] il_operand]

    il_fb_call ::= (’CAL’ | ’CALC’ | ’CALCN’) fb_name ’(’
    il_operand_list ’)’
    ```
  - **B.2.2:**
    ```
    il_operator ::= ’LD’ | ’LDN’ | ’ST’ | ’STN’ | ’S’ | ’R’
    | (’AND’ | ’ANDN’ | ’OR’ | ’ORN’ | ’XOR’ | ’XORN’) [’(’]
    | (’ADD’ | ’SUB’ | ’MUL’ | ’DIV’) [’(’]
    | (’GT’ | ’GE’ | ’EQ’ | ’NE’ | ’LT’ | ’LE’) [’(’]
    | ’JMP’ | ’JMPC’ | ’JMPCN’ | ’)’ | function_name
    ```

Configuration examples

E

Overview

This section contains various configuration examples, given as step-by-step instructions.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
E.1	Quantum Example - Remote Control with RIO	878
E.2	Quantum Example - Remote control with RIO (series 800)	887
E.3	Quantum Example - Remote Control with DIO	900
E.4	Quantum Example – INTERBUS Control	910
E.5	Quantum Example - SY/MAX Controller	916
E.6	Quantum Example - Profibus DP Controller	925
E.7	Quantum-Example - Peer Cop	941
E.8	Compact Example	950
E.9	Atrium Example – INTERBUS Controller	955
E.10	Momentum Example - Remote I/O Bus	965
E.11	Momentum Example - Ethernet Bus System	974

E.1 Quantum Example - Remote Control with RIO

Overview

This Chapter contains the step-by-step process for the configuration of remote control with RIO (**R**emote **I**/**O**).

What's in this Section?

This section contains the following topics:

Topic	Page
Editing local drop	879
Editing Remote Drop	884

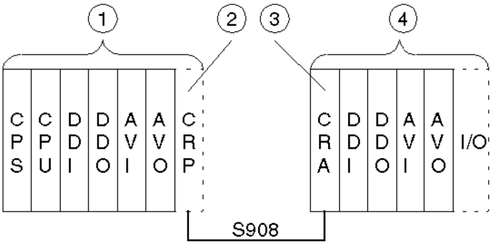
Editing local drop

Introduction

This section describes the configuration of the first (local) drop. The processing sequence begins first of all with the definition of all drops.

When editing the first (local) drop the modules must be set with their I/O references before the individual modules can be parameterized.

Quantum – remote controller with RIO

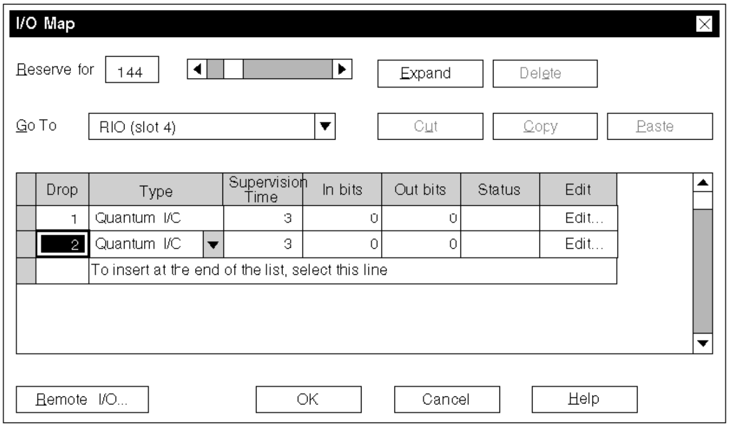


1 Local Quantum drop 1
2 RIO master module
3 RIO slave module
4 RIO drop 2

Defining Drops

To define drops proceed as follows in the **PLC Configuration** window:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Select <b>PLC Selection</b>.
<b>Response:</b> The <b>PLC selection</b> dialog is opened.</td></tr>
<tr><td>2</td><td>Select the <b>Quantum</b> PLC family and a <b>CPU x113 xx</b>. Using <b>OK</b> return to the <b>PLC Configuration</b> window.</td></tr>
<tr><td>3</td><td>Select <b>I/O Map</b>.
<b>Response:</b> The <b>I/O Map</b> dialog is opened and the first drop is automatically entered in the table.</td></tr>
<tr><td>4</td><td>Select the <b>Head Setup</b> command button.
<b>Response:</b> The <b>Head Setup</b> dialog is opened.</td></tr>
<tr><td>5</td><td>Enter a 7 in <b>RIO Slot</b> and quit the dialog using <b>OK</b>.
<b>Response:</b> The CRP-93x-00 module is automatically inserted in the component list (in slot 7) of the selected drop. In the <b>Go To</b> list box, the <b>Local/RIO (Slot 7)</b> network link is displayed.</td></tr>
<tr><td>6</td><td>Select the last line in the table.
Select the <b>Insert</b> command button.
<b>Response:</b> The second drop is entered in the <b>Type</b> column.
<b>Note:</b> The number of drops to be inserted is defined in the <b>segment scheduler</b> dialog. The default predetermines a maximum number of 32.
Dialog display



I/O Map

Reserve for 144

Go To RIO (slot 4)

<table>
<tr><th>Drop</th><th>Type</th><th>Supervision Time</th><th>In bits</th><th>Out bits</th><th>Status</th><th>Edit</th></tr>
<tr><td>1</td><td>Quantum I/C</td><td>3</td><td>0</td><td>0</td><td></td><td>Edit...</td></tr>
<tr><td>2</td><td>Quantum I/C</td><td>3</td><td>0</td><td>0</td><td></td><td>Edit...</td></tr>
<tr><td></td><td colspan="6">To insert at the end of the list, select this line</td></tr>
</table>

</td></tr>
<tr><td>7</td><td>Select the drop from the <b>Drop</b> column.
Select the <b>Edit...</b> command button.
<b>Response:</b> You reach the module map.</td></tr>
</table>

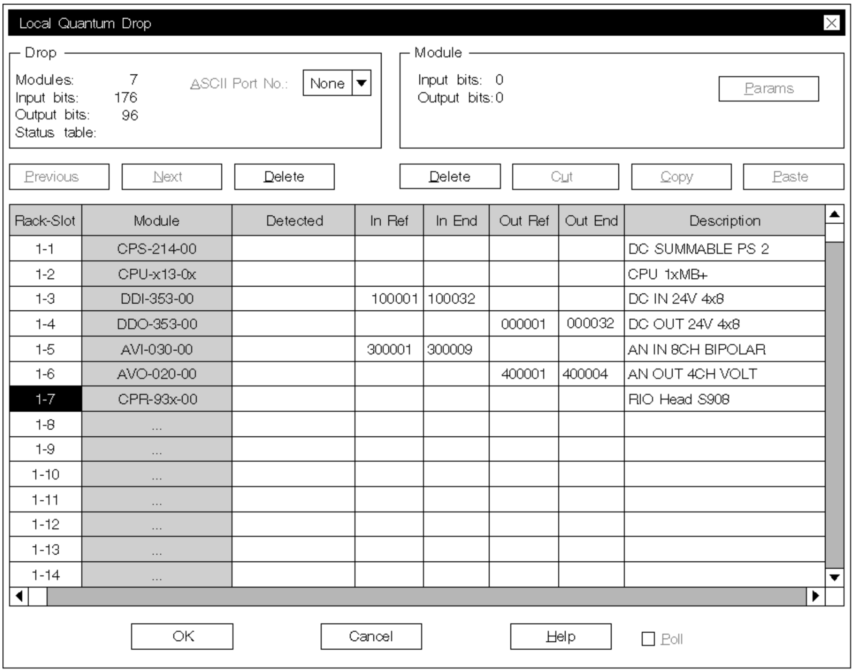
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column, select the **CPS-214-00** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see **Local Quantum Drop** dialog representation).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



Modules: 7
Input bits: 176
Output bits: 96
Status table:

ASCII Port No.: None

Input bits: 0
Output bits: 0

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1-1	CPS-214-00						DC SUMMABLE PS 2
1-2	CPU-x13-0x						CPU 1xMB+
1-3	DDI-353-00		100001	100032			DC IN 24V 4x8
1-4	DDO-353-00				000001	000032	DC OUT 24V 4x8
1-5	AVI-030-00		300001	300009			AN IN 8CH BIPOLAR
1-6	AVO-020-00				400001	400004	AN OUT 4CH VOLT
1-7	CPR-93x-00						RIO Head S908
1-8	...						
1-9	...						
1-10	...						
1-11	...						
1-12	...						
1-13	...						
1-14	...						

Set module parameters

To set parameters for the individual modules, proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	From the **Rack Slot** column select the **1-3** line. **Response:** The **1-3** text box has a dark background, i.e. the **DDI-353-00** module has been selected for editing. **Note:** Parameters are not set for the **CPS-214-00** and **CPU-x13-0x** modules.
2	Select the **Params** command button. **Response:** The **140-DDI-353-00** dialog is opened.
3	Select the **Discrete** option button. **Response:** You return to the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example. **Note:** The modules are sometimes set with different parameters. Help with this can be obtained from the corresponding help texts in the parameter dialog.

Editing Remote Drop

Editing of the I/O st. (Drop) defined second takes place in the dialog **RIO (Slot 7) - quantum I/O-St. 2**.

This dialog can be reached in two ways:

- In the **I/O Map** dialog, using the **Edit....** command button, or
- in the **Local Quantum Drop** dialog, using the **Next** command button.

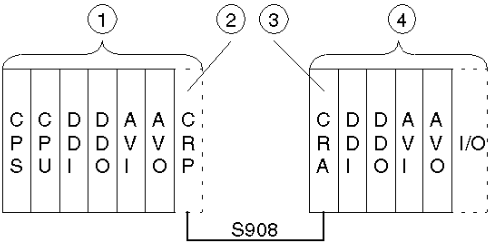
Editing Remote Drop

Introduction

This section describes the configuration of the second (remote) drop. The drop has already been defined in Editing the First (local) Drop *(see page 879)*.

To edit the second (remote) drop, the modules must be specified with their I/O references before parameters for the individual modules can be set.

Quantum – remote controller with RIO



1 Local Quantum drop 1
2 RIO master module
3 RIO slave module
4 RIO drop 2

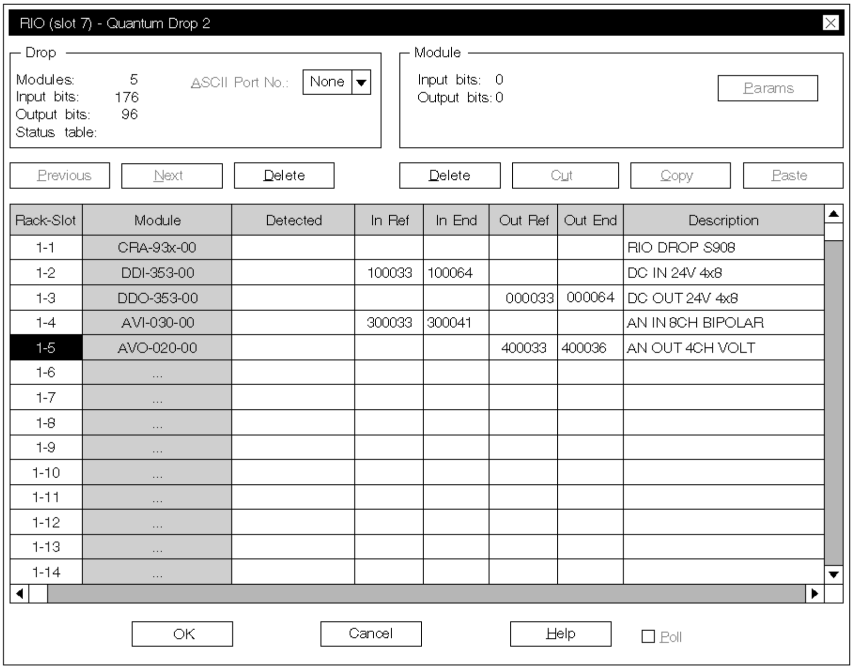
Mapping Modules and Specifying I/O References

To allocate the modules and specify the address ranges use the dialog **RIO (slot 7) - quantum I/O-St. 2** and proceed as follows:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column select the **CRA-93x-00** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see dialog representation **RIO (Slot 7) Quantum Drop 2**).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



RIO (slot 7) - Quantum Drop 2

Drop

Modules: 5
Input bits: 176
Output bits: 96
Status table:

ASCII Port No.: None

Module

Input bits: 0
Output bits: 0

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1-1	CRA-93x-00						RIO DROP S908
1-2	DDI-353-00		100033	100064			DC IN 24V 4x8
1-3	DDO-353-00				000033	000064	DC OUT 24V 4x8
1-4	AVI-030-00		300033	300041			AN IN 8CH BIPOLAR
1-5	AVO-020-00				400033	400036	AN OUT 4CH VOLT
1-6	...						
1-7	...						
1-8	...						
1-9	...						
1-10	...						
1-11	...						
1-12	...						
1-13	...						
1-14	...						

Set module parameters

To set parameters for individual modules use the dialog **RIO (slot 7) - Quantum I/O-St. 2** and proceed as follows:

Step	Action
1	From the **Rack Slot** column select the **1-2** line. **Response:** The **1-2** text box has a dark background, i.e. the **DDI-353-00** module has been selected for editing.
2	Select the **Params** command button. **Response:** The **140-DDI-353-00** dialog is opened.
3	Select the **Discrete** option button. **Response:** You return to the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example. **Note:** The modules are sometimes set with different parameters. Help with this can be obtained from the corresponding help texts in the parameter dialog.

E.2 Quantum Example - Remote control with RIO (series 800)

Overview

This Chapter contains the step-by-step process for the configuration of remote control with RIO (**R**emote **I**/**O**) and series 800 modules.

What's in this Section?

This section contains the following topics:

Topic	Page
Editing Local Drop	888
Editing Remote Drop	893
Editing Remote Drop	897

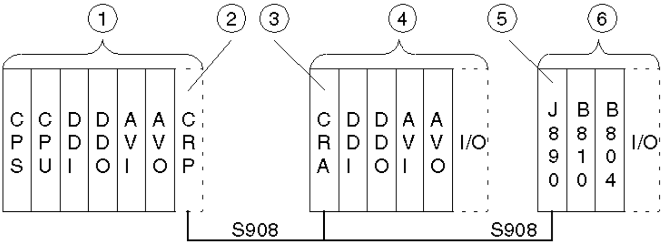
Editing Local Drop

Introduction

This section describes the configuration of the first (local) drop. The processing sequence begins first of all with the definition of all drops.

When editing the first (local) drop the modules must be set with their I/O references before parameters can be set forindividual modules.

Quantum – remote controller with RIO (Series 800)

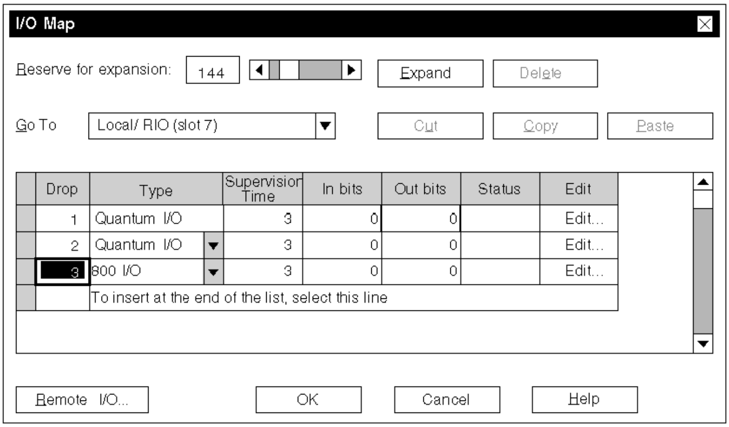


1 Local Quantum drop 1
2 RIO master module
3 RIO slave module
4 RIO drop 2
5 Adapter module
6 RIO drop 3 with series 800 modules

Defining Drops

To define drops proceed as follows in the **PLC Configuration** window:

Step	Action
1	Select **PLC Selection**. **Response:** The **PLC selection** dialog is opened.
2	Select the **Quantum** PLC family and a **CPU x113 xx**. Using **OK** return to the **PLC Configuration** window.
3	Select **I/O Map**. **Response:** The **I/O Map** dialog is opened and the first drop is automatically entered in the table.
4	Select the **Head Setup** command button. **Response:** The **Head Setup** dialog is opened.
5	Enter a 7 in **RIO Slot** and quit the dialog using **OK**. **Response:** The CRP-93x-00 module is automatically inserted in the I/O map (in slot 7) of the selected drop. In the **Go To** list box, the **Local/RIO (Slot 7)** network link is displayed.

Step	Action
6	Select the last free row in the table, and insert the second drop with the command button **Insert**. **Response:** The second drop is entered in the **Type** column of the table. **Note:** The number of drops to be inserted is defined in the **segment scheduler** dialog. The default predetermines the maximum number of 32, so that settings are not necessary.
7	Select the last free row in the table again, and insert the third drop with the **Insert** command button. **Response:** The second drop is entered in the **Type** column of the table.
8	Select the third drop and open the list box in the **Type** column. Select the **800 I/O** option. Dialog display  I/O Map Expand Delete Cut Copy Paste Remote I/O... OK Cancel Help Reserve for expansion: 144 Go To: Local/ RIO (slot 7) Drop | Type | Supervision Time | In bits | Out bits | Status | Edit 1 | Quantum I/O | 3 | 0 | 0 | | Edit... 2 | Quantum I/O | 3 | 0 | 0 | | Edit... 3 | 800 I/O | 3 | 0 | 0 | | Edit... To insert at the end of the list, select this line
9	Select the first drop from the **Drop** column. Select the **Edit...** command button. **Response:** You reach the module map.

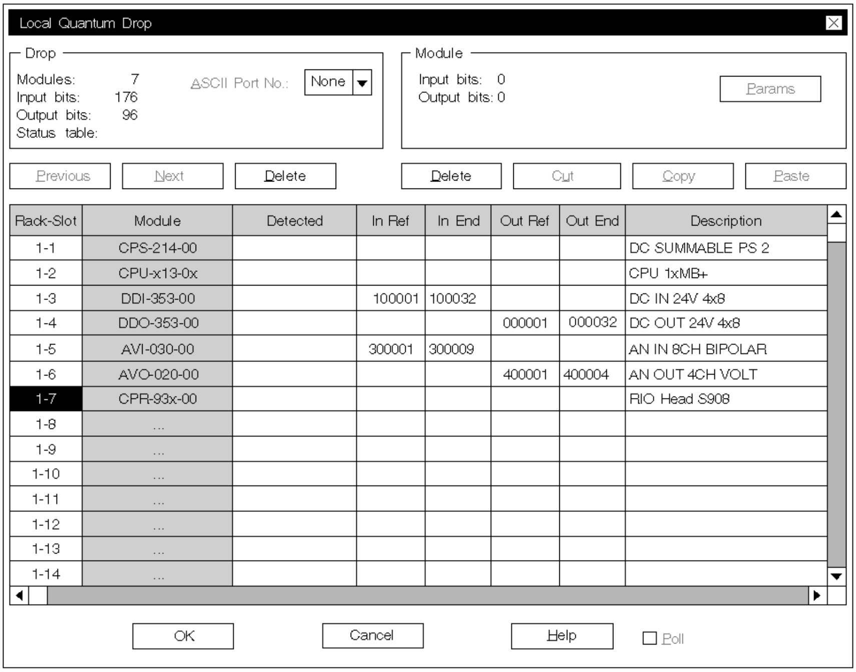
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column, select the **CPS-214-00** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see **Local Quantum Drop** dialog representation).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



Modules: 7
Input bits: 176
Output bits: 96
Status table:
ASCII Port No.: None

Input bits: 0
Output bits: 0

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1-1	CPS-214-00						DC SUMMABLE PS 2
1-2	CPU-x13-0x						CPU 1xMB+
1-3	DDI-353-00		100001	100032			DC IN 24V 4x8
1-4	DDO-353-00				000001	000032	DC OUT 24V 4x8
1-5	AVI-030-00		300001	300009			AN IN 8CH BIPOLAR
1-6	AVO-020-00				400001	400004	AN OUT 4CH VOLT
1-7	CPR-93x-00						RIO Head S908
1-8	...						
1-9	...						
1-10	...						
1-11	...						
1-12	...						
1-13	...						
1-14	...						

Set module parameters

To set parameters for the individual modules, proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	From the **Rack Slot** column select the **1-3** line. **Response:** The **1-3** text box has a dark background, i.e. the **DDI-353-00** module has been selected for editing. **Note:** Parameters are not set for the **CPS-214-00** and **CPU-x13-0x** modules.
2	Select the **Params** command button. **Response:** The **140-DDI-353-00** dialog is opened.
3	Select the **Discrete** option button. **Response:** You return to the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example. **Note:** The modules are sometimes set with different parameters. Help with this can be obtained from the corresponding help texts in the parameter dialog.

Editing Remote Drop

Editing of the I/O st. (Drop) defined second takes place in the dialog **RIO (Slot 7) - quantum I/O-St. 2**.

This dialog can be reached in two ways:

- In the **I/O Map** dialog, using the **Edit....** command button, or
- in the **Local Quantum Drop** dialog, using the **Next** command button.

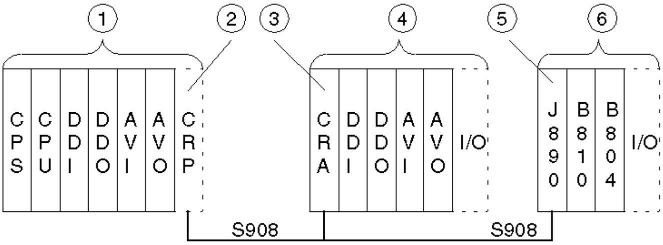
Editing Remote Drop

Introduction

This section describes the configuration of the second (remote) drop. The drop has already been defined in Editing the First (local) Drop *(see page 888)*.

To edit the second (remote) drop, the modules must be specified with their I/O references before parameters for the individual modules can be set.

Quantum – remote controller with RIO (Series 800)



1 Local Quantum drop 1
2 RIO master module
3 RIO slave module
4 RIO drop 2
5 Adapter module
6 RIO drop 3 with series 800 modules

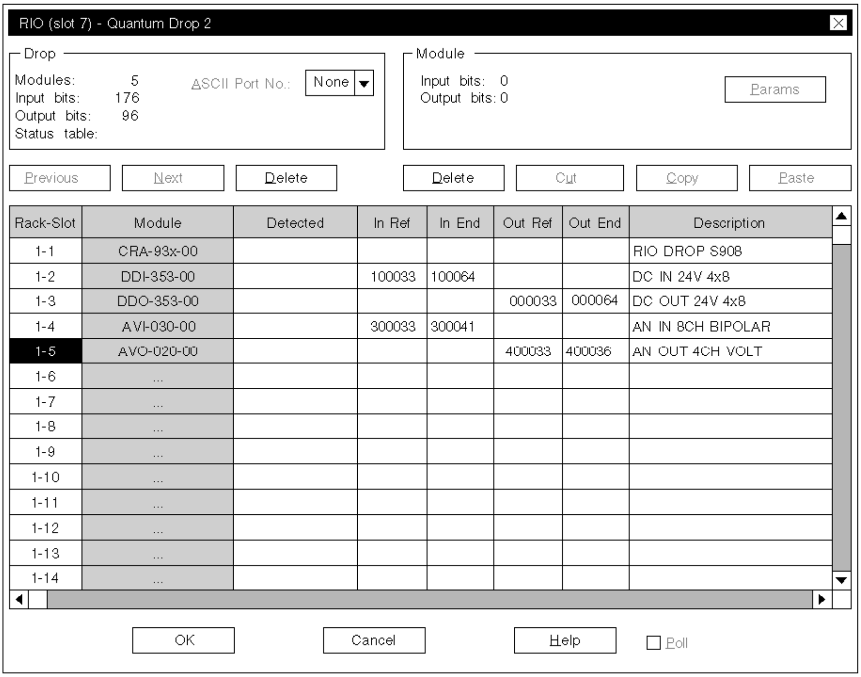
Mapping Modules and Specifying I/O References

To allocate the modules and specify the address ranges use the dialog **RIO (slot 7) - quantum I/O-St. 2** and proceed as follows:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column select the **CRA-93x-00** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see dialog representation **RIO (Slot 7) Quantum Drop 2**).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



Modules: 5
Input bits: 176
Output bits: 96
Status table:

Input bits: 0
Output bits: 0

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1-1	CRA-93x-00						RIO DROP S908
1-2	DDI-353-00		100033	100064			DC IN 24V 4x8
1-3	DDO-353-00				000033	000064	DC OUT 24V 4x8
1-4	AVI-030-00		300033	300041			AN IN 8CH BIPOLAR
1-5	AVO-020-00				400033	400036	AN OUT 4CH VOLT
1-6	...						
1-7	...						
1-8	...						
1-9	...						
1-10	...						
1-11	...						
1-12	...						
1-13	...						
1-14	...						

Set module parameters

To set parameters for individual modules use the dialog **RIO (slot 7) - Quantum I/O-St. 2** and proceed as follows:

Step	Action
1	From the **Rack Slot** column select the **1-2** line. **Response:** The **1-2** text box has a dark background, i.e. the **DDI-353-00** module has been selected for editing.
2	Select the **Params** command button. **Response:** The **140-DDI-353-00** dialog is opened.
3	Select the **Discrete** option button. **Response:** You return to the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example. **Note:** The modules are sometimes set with different parameters. Help with this can be obtained from the corresponding help texts in the parameter dialog.

Editing Remote 800 Drops

The third defined drop is edited in the **RIO (Slot 7) - 800 Drop 3** dialog.

This dialog can be reached in two ways:

- In the **I/O Map** dialog, using the **Edit....** command button, or
- in the **RIO (Slot 7) - Quantum Drop 2** dialog using the **Next** command button.

Editing Remote Drop

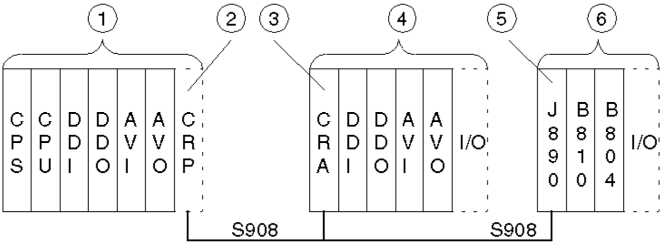
Introduction

This Section describes the configuration of the third (remote) drop. The drop has already been defined in Editing the First (local) Drop *(see page 888)*.

To edit the third (remote) drop, the modules must be specified with their I/O references before the individual modules can be parameterized.

NOTE: The J890 adapter module must be mounted in the rack of the third drop. However, this module is not visible either in the software or in the dialogs.

Quantum – remote controller with RIO (Series 800)



1 Local Quantum drop 1
2 RIO master module
3 RIO slave module
4 RIO drop 2
5 Adapter module
6 RIO drop 3 with series 800 modules

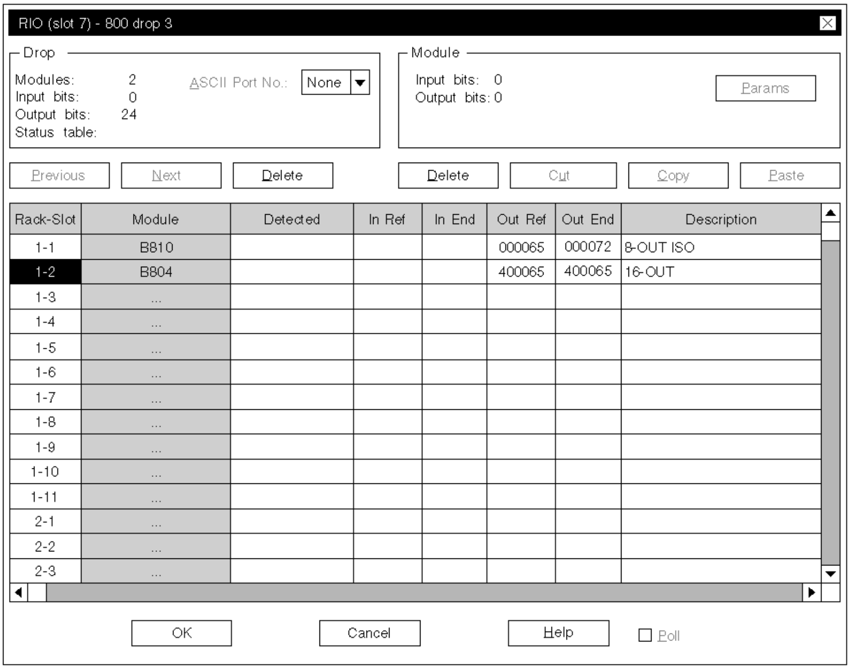
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges go to the **RIO (slot 7) - 800 drop 3** dialog and proceed as follows:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column select the **B810** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see dialog representation **RIO (slot 7) 800 drop 3**).
5	In the **Out Ref.** column, enter the start references for the output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference of the available address range (**Out End** column) is entered automatically.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



RIO (slot 7) - 800 drop 3

Drop

Modules: 2
Input bits: 0
Output bits: 24
Status table:

ASCII Port No.: None

Module

Input bits: 0
Output bits: 0

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1-1	B810				000065	000072	8-OUT ISO
1-2	B804				400065	400065	16-OUT
1-3	...						
1-4	...						
1-5	...						
1-6	...						
1-7	...						
1-8	...						
1-9	...						
1-10	...						
1-11	...						
2-1	...						
2-2	...						
2-3	...						

Set module parameters

To set parameters for individual modules go to the **RIO (slot 7) - 800 drop 3** dialog and proceed as follows:

Step	Action
1	From the **Rack Slot** column select the **1-1** line. **Response:** The **1-1** text box has a dark background, i.e. the **B810** module has been selected for editing.
2	Select the **Params** command button. **Response:** The **B810** dialog is opened.
3	Select the option button **Discrete** **Response:** You return to the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example. **Note:** The modules are sometimes set with different parameters. Help with this can be obtained from the corresponding help texts in the parameter dialog.

E.3 Quantum Example - Remote Control with DIO

Overview

This Chapter contains the step-by-step process for the configuration of remote control with DIO (**D**istributed **I**/**O**).

What's in this Section?

This section contains the following topics:

Topic	Page
Editing Local Drop	901
Editing Local Drop	906

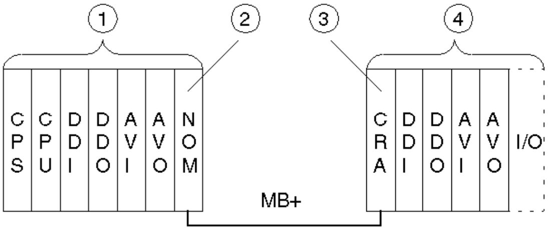
Editing Local Drop

Introduction

This section describes the configuration of the first (local) drop. The processing sequence begins first of all with the definition of the drop.

When editing the first (local) drop the modules must be set with their I/O references before parameters can be set forindividual modules.

Quantum – remote controller with DIO

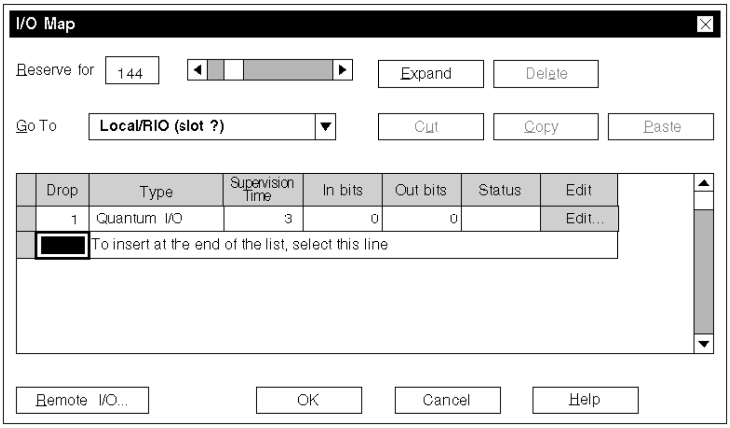


1 Local Quantum drop 1
2 DIO master module
3 DIO slave module
4 DIO drop 2

Defining the Drop

To define the drop use **Configure** from the main menu and proceed as follows:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Select <b>PLC Selection</b>.
<b>Response:</b> The <b>PLC selection</b> dialog is opened.</td></tr>
<tr><td>2</td><td>Select the <b>Quantum</b> PLC family and a <b>CPU x113 xx</b>. Using <b>OK</b> return to the <b>PLC Configuration</b> window.</td></tr>
<tr><td>3</td><td>Select <b>I/O Map</b>.
<b>Response:</b> The <b>I/O Map</b> dialog is opened and the first drop is automatically entered in the table.</td></tr>
<tr><td>4</td><td>Select the <b>Head Setup</b> command button.
<b>Response:</b> The <b>Head Setup</b> dialog is opened.</td></tr>
<tr><td>5</td><td>Enter a 7 in <b>NOM Slot 1</b> and quit the dialog using <b>OK</b>.
<b>Response:</b> The NOM module NOM-2xx-00 is automatically inserted in the I/O map (in slot 7) of the selected drop. In the <b>Go to</b> list box, the network link <b>Local/RIO (Slot ?)</b> is displayed.
Dialog display



I/O Map

Reserve for 144

Go To Local/RIO (slot ?)

<table>
<tr><th>Drop</th><th>Type</th><th>Supervision Time</th><th>In bits</th><th>Out bits</th><th>Status</th><th>Edit</th></tr>
<tr><td>1</td><td>Quantum I/O</td><td>3</td><td>0</td><td>0</td><td></td><td>Edit...</td></tr>
<tr><td></td><td colspan="6">To insert at the end of the list, select this line</td></tr>
</table>

</td></tr>
<tr><td>6</td><td>Select the drop from the <b>Drop</b> column.
Select the <b>Edit...</b> command button.
<b>Response:</b> You reach the module map.</td></tr>
</table>

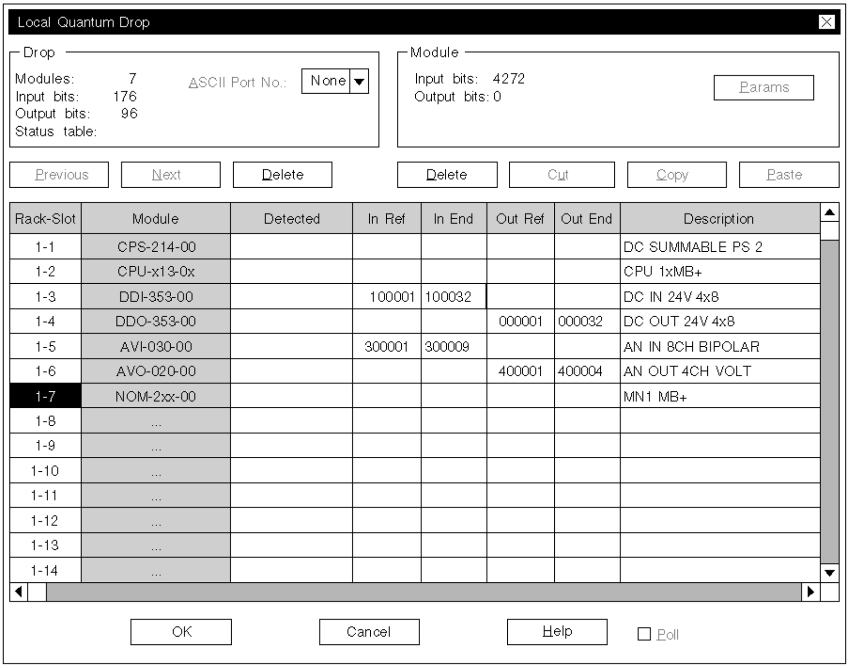
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column, select the **CPS-214-00** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see **Local Quantum Drop** dialog representation).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



Local Quantum Drop

Drop

Modules: 7
Input bits: 176
Output bits: 96
Status table:

ASCII Port No.: None

Module

Input bits: 4272
Output bits: 0

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1-1	CPS-214-00						DC SUMMABLE PS 2
1-2	CPU-x13-0x						CPU 1xMB+
1-3	DDI-353-00		100001	100032			DC IN 24V 4x8
1-4	DDO-353-00				000001	000032	DC OUT 24V 4x8
1-5	AVI-030-00		300001	300009			AN IN 8CH BIPOLAR
1-6	AVO-020-00				400001	400004	AN OUT 4CH VOLT
1-7	NOM-2xx-00						MN1 MB+
1-8	...						
1-9	...						
1-10	...						
1-11	...						
1-12	...						
1-13	...						
1-14	...						

Set module parameters

To set parameters for the individual modules, proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	From the **Rack Slot** column select the **1-3** line. **Response:** The **1-3** text box has a dark background, i.e. the **DDI-353-00** module has been selected for editing. **Note:** Parameters are not set for the **CPS-214-00** and **CPU-x13-0x** modules.
2	Select the **Params** command button. **Response:** The **140-DDI-353-00** dialog is opened.
3	Select the **Discrete** option button. **Response:** You return to the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example. **Note:** The modules are sometimes set with different parameters. Help with this can be obtained from the corresponding help texts in the parameter dialog.

Editing Remote Drop

To edit the remote drop with DIO, you must return to the **I/O Map** dialog and define the drop.

Editing Local Drop

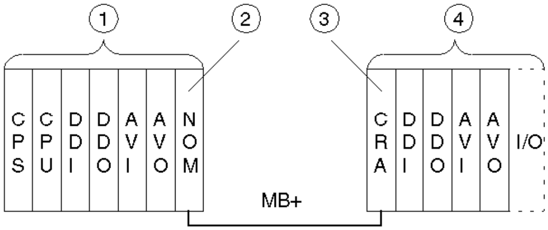
Introduction

This section describes the configuration of the second (remote) drop. The processing sequence begins first of all with the definition of the drop.

To edit the second (remote) drop, the modules must be specified with their I/O references before parameters for the individual modules can be set.

NOTE: To link to the remote network, the coupling module CRA-21x-x0 must be entered during module mapping.

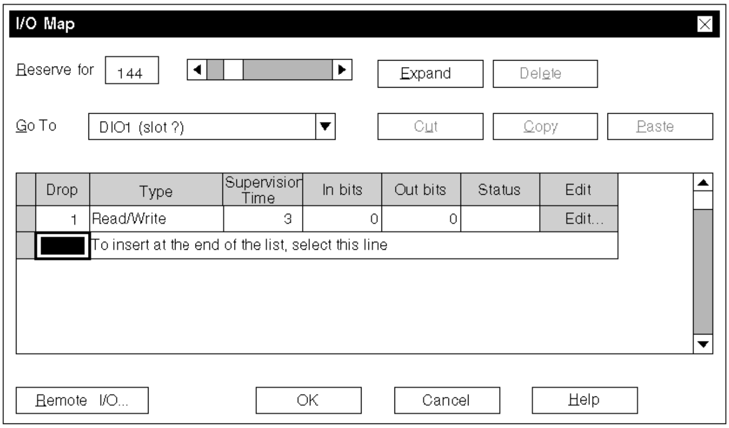
Quantum – remote controller with DIO



1 Local Quantum drop 1
2 DIO master module
3 DIO slave module
4 DIO drop 2

Defining the Drop

To define the drop go to the **I/O map** dialog and proceed as follows:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>From the <b>Go to</b> list box, select the <b>DIO 1 (Slot 7)</b> network link.
<b>Response:</b> The drop entered in the table is no longer displayed.</td></tr>
<tr><td>2</td><td>Select the <b>Insert</b> command button.
<b>Response:</b> In the <b>Type</b> column, the <b>Read/Write</b> type is entered.
Dialog display

Reserve for 144
Go To DIO1 (slot ?)
<table>
<tr><th></th><th>Drop</th><th>Type</th><th>Supervision Time</th><th>In bits</th><th>Out bits</th><th>Status</th><th>Edit</th></tr>
<tr><td></td><td>1</td><td>Read/Write</td><td>3</td><td>0</td><td>0</td><td></td><td>Edit...</td></tr>
<tr><td></td><td colspan="7">To insert at the end of the list, select this line</td></tr>
</table>
I/O Map
Expand
Delete
Cut
Copy
Paste
Remote I/O...
OK
Cancel
Help

</td></tr>
<tr><td>3</td><td>Select the drop from the <b>Drop</b> column.
Select the <b>Edit...</b> command button.
<b>Response:</b> You reach the module map.</td></tr>
</table>

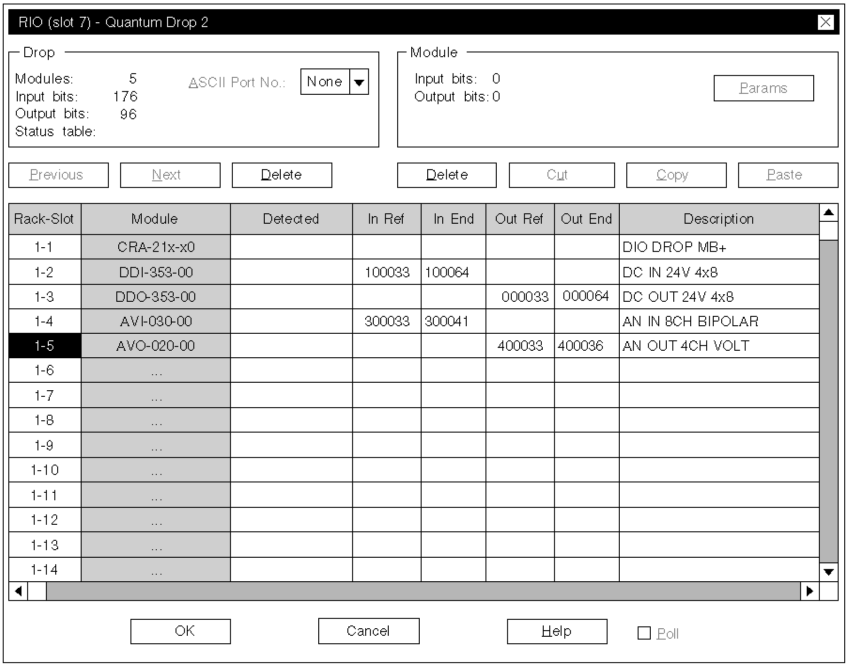
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges use the **DIO 1 (slot 7) - drop 1** dialog and proceed as follows:

Step	Action
1	Select the **Module** →**...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column select the **CRA-21x-x0** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see dialog representation **DIO (slot 7) Quantum drop 1**).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



RIO (slot 7) - Quantum Drop 2

Drop

Modules: 5
Input bits: 176
Output bits: 96
Status table:

ASCII Port No.: None

Module

Input bits: 0
Output bits: 0

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1-1	CRA-21x-x0						DIO DROP MB+
1-2	DDI-353-00		100033	100064			DC IN 24V 4x8
1-3	DDO-353-00				000033	000064	DC OUT 24V 4x8
1-4	AVI-030-00		300033	300041			AN IN 8CH BIPOLAR
1-5	AVO-020-00				400033	400036	AN OUT 4CH VOLT
1-6	...						
1-7	...						
1-8	...						
1-9	...						
1-10	...						
1-11	...						
1-12	...						
1-13	...						
1-14	...						

Set module parameters

To set parameters for individual modules use the **DIO 1 (slot 7) - Drop 1** dialog and proceed as follows:

Step	Action
1	From the **Rack Slot** column select the **1-2** line. **Response:** The **1-2** text box has a dark background, i.e. the **DDI-353-00** module has been selected for editing.
2	Select the **Params** command button. **Response:** The **140-DDI-353-00** dialog is opened.
3	Select the **Discrete** option button. **Response:** You return to the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example. **Note:** The modules are sometimes set with different parameters. Help with this can be obtained from the corresponding help texts in the parameter dialog.

E.4 Quantum Example – INTERBUS Control

Overview

This Chapter contains the step-by-step process for the configuration of INTERBUS control with the Quantum.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information	911
Editing Local Drop	912

General Information

Introduction

INTERBUS control using Quantum occurs via module NOA-611-10. During this process the module collects the words of all remote bus nodes and creates a telegram with status information and I/O words. The telegram is then transferred to the CPU, so that the NOA behaves like an I/O module.

NOTE: Using branch interfaces in the remote bus, remote bus branches with further remote bus nodes (TIOs) can be constructed. However, the branch interfaces can only be inserted in the remote bus, not in the remote bus branch.

Parameterization

Command sequence parameterization (restart procedure) occurs in the CMD Tool, produced by the PHÖNIX firm (see also " NOA 611 1 restart procedure" with an example for parameterizing the command sequence in CMD Tool).

Editing Local Drop

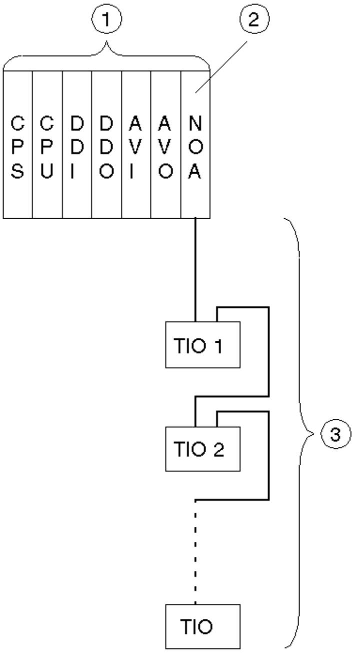
Introduction

This section describes the configuration of the first (local) drop. The processing sequence begins first of all with the definition of the drop.

When editing the first (local) drop the modules must be set with their I/O references before parameters can be set for individual modules.

NOTE: When the NOA-611-00 module is entered in the I/O map, the loadable ULEX is automatically installed.

Quantum - INTERBUS controller

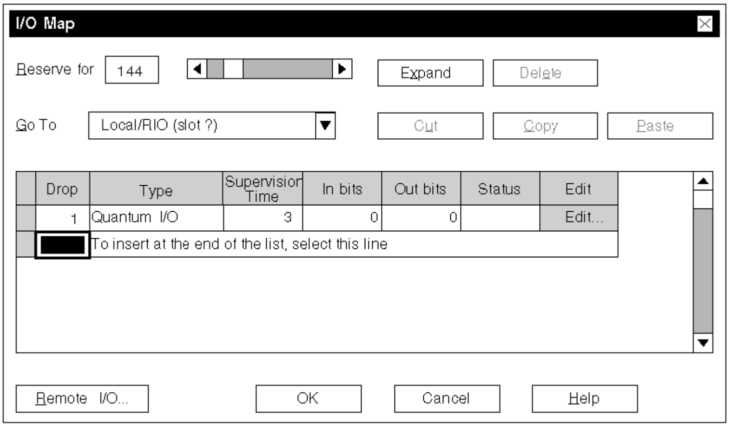


1 Local Quantum Drop
2 INTERBUS master module
3 Remote bus without branch interface

NOTE: The configuration of remote bus nodes does not take place in Concept and is therefore not apparent in the I/O map. To edit the remote bus nodes, you must use the CMD tool produced by the PHÖNIX firm (**C**onfiguration **M**onitoring and **D**iagnostic Software).

Defining Drops

To define drops proceed as follows in the **PLC Configuration** window:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Select <b>PLC Selection</b>.
<b>Response:</b> The <b>PLC selection</b> dialog is opened.</td></tr>
<tr><td>2</td><td>Select the <b>Quantum</b> PLC family and a <b>CPU x113 xx</b>. Use <b>OK</b> return to the <b>PLC Configuration</b> window.</td></tr>
<tr><td>3</td><td>Select <b>I/O Map</b>.
<b>Response:</b> The <b>I/O Map</b> dialog is opened and the first drop is automatically entered in the table.
Dialog display


Reserve for 144
Go To Local/RIO (slot ?)
<table>
<tr><th>Drop</th><th>Type</th><th>Supervision Time</th><th>In bits</th><th>Out bits</th><th>Status</th><th>Edit</th></tr>
<tr><td>1</td><td>Quantum I/O</td><td>3</td><td>0</td><td>0</td><td></td><td>Edit...</td></tr>
<tr><td>■</td><td colspan="6">To insert at the end of the list, select this line</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>Select the drop from the <b>Drop</b> column.
Select the <b>Edit...</b> command button.
<b>Response:</b> You reach the module map.</td></tr>
</table>

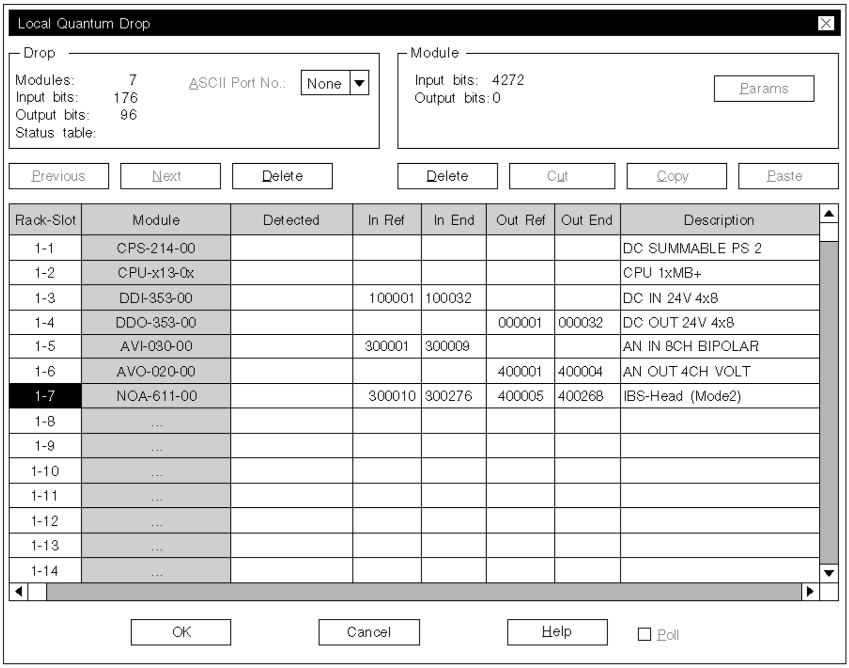
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column, select the **CPS-214-00** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see **Local Quantum Drop** dialog representation).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



Drop
Modules: 7
Input bits: 176
Output bits: 96
Status table:

Module
Input bits: 4272
Output bits: 0

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1-1	CPS-214-00						DC SUMMABLE PS 2
1-2	CPU-x13-0x						CPU 1xMB+
1-3	DDI-353-00		100001	100032			DC IN 24V 4x8
1-4	DDO-353-00				000001	000032	DC OUT 24V 4x8
1-5	AVI-030-00		300001	300009			AN IN 8CH BIPOLAR
1-6	AVO-020-00				400001	400004	AN OUT 4CH VOLT
1-7	NOA-611-00		300010	300276	400005	400268	IBS-Head (Mode2)
1-8	...						
1-9	...						
1-10	...						
1-11	...						
1-12	...						
1-13	...						
1-14	...						

Set module parameters

To set parameters for the individual modules, proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	From the **Rack Slot** column select the **1-3** line. **Response:** The **1-3** text box has a dark background, i.e. the **DDI-353-00** module has been selected for editing. **Note:** Parameters are not set for the **CPS-214-00** and **CPU-x13-0x** modules.
2	Select the **Params** command button. **Response:** The **140-DDI-353-00** dialog is opened.
3	Select the **Discrete** option button. **Response:** You return to the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example. **Note:** The modules are sometimes set with different parameters. Help with this can be obtained from the corresponding help texts in the parameter dialog.

E.5 Quantum Example - SY/MAX Controller

Overview

This Chapter contains the step-by-step process for the configuration of a SY/MAX controller.

What's in this Section?

This section contains the following topics:

Topic	Page
Editing Local Drop	917
Editing Remote Drop	922

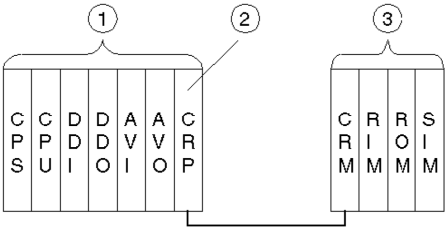
Editing Local Drop

Introduction

This section describes the configuration of the first (local) drop. The processing sequence begins first of all with the definition of all drops.

When editing the first (local) drop the modules must be set with their I/O references before parameters can be set forindividual modules.

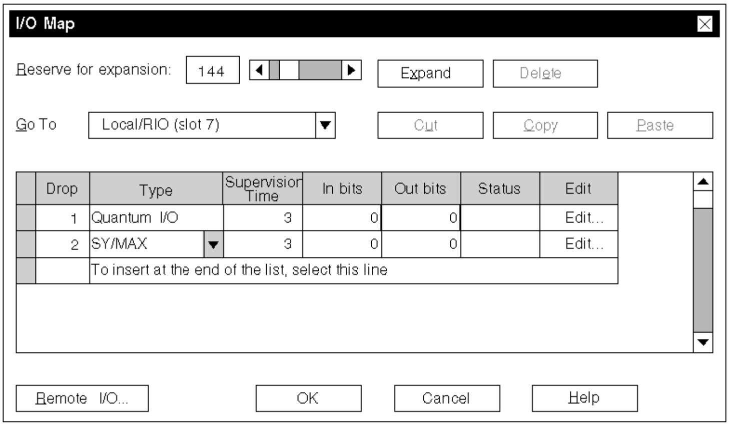
Quantum – SY/MAX controller



1 Local Quantum drop 1
2 RIO master module
3 SY/MAX drop 2

Defining Drops

To define drops proceed as follows in the **PLC Configuration** window:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Select <b>PLC Selection</b>.
<b>Response:</b> The <b>PLC selection</b> dialog is opened.</td></tr>
<tr><td>2</td><td>Select the <b>Quantum</b> PLC family and a <b>CPU x113 xx</b>. Using <b>OK</b> return to the <b>PLC Configuration</b> window.</td></tr>
<tr><td>3</td><td>Select <b>I/O Map</b>.
<b>Response:</b> The <b>I/O Map</b> dialog is opened and the first drop is automatically entered in the table.</td></tr>
<tr><td>4</td><td>Select the <b>Head Setup</b> command button.
<b>Response:</b> The <b>Head Setup</b> dialog is opened.</td></tr>
<tr><td>5</td><td>Enter a 7 in <b>RIO Slot</b> and quit the dialog using <b>OK</b>.
<b>Response:</b> The CRP-93x-00 module is automatically inserted in the I/O map (in slot 7) of the selected drop. In the <b>Go To</b> list box, the <b>Local/RIO (Slot 7)</b> network link is displayed.</td></tr>
<tr><td>6</td><td>Select the last line in the table.
Select the <b>Insert</b> command button.
<b>Response:</b> The second drop is entered in the <b>Type</b> column.
<b>Note:</b> The number of drops to be inserted is defined in the <b>segment scheduler</b> dialog. The default defines a maximum number of 32.</td></tr>
<tr><td>7</td><td>Select the second drop and in the <b>Type</b> column, open the list box.
Select the <b>SY/MAX</b>option.
Dialog display

I/O Map
Reserve for expansion: 144
Go To: Local/RIO (slot 7)
<table>
<tr><th>Drop</th><th>Type</th><th>Supervision Time</th><th>In bits</th><th>Out bits</th><th>Status</th><th>Edit</th></tr>
<tr><td>1</td><td>Quantum I/O</td><td>3</td><td>0</td><td>0</td><td></td><td>Edit...</td></tr>
<tr><td>2</td><td>SY/MAX</td><td>3</td><td>0</td><td>0</td><td></td><td>Edit...</td></tr>
<tr><td></td><td>To insert at the end of the list, select this line</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
Expand
Delete
Cut
Copy
Paste
Remote I/O...
OK
Cancel
Help

</td></tr>
<tr><td>8</td><td>Select the first drop from the <b>Drop</b> column.
Select the <b>Edit...</b> command button.
<b>Response:</b> You reach the module map.</td></tr>
</table>

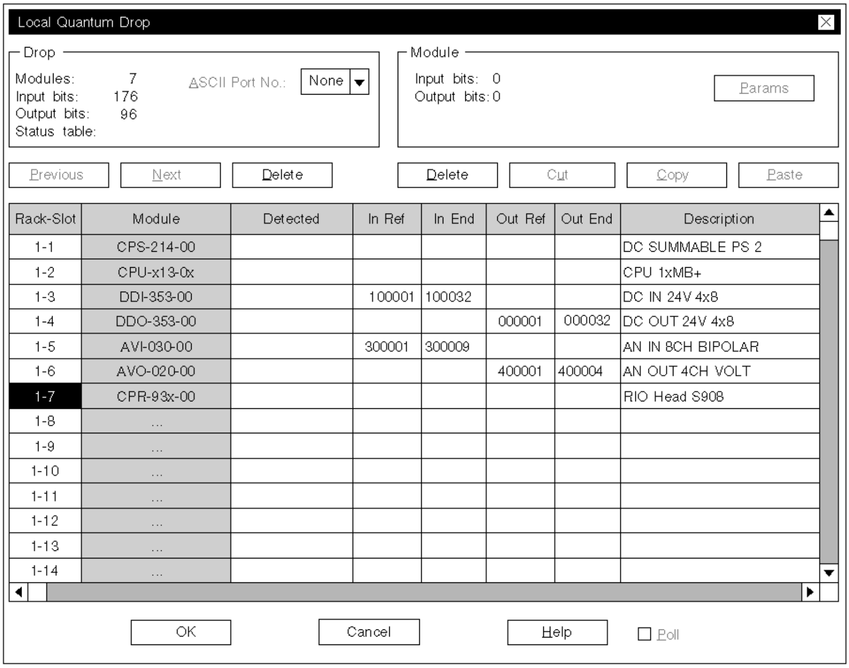
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column, select the **CPS-214-00** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see **Local Quantum Drop** dialog representation).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



Modules: 7
Input bits: 176
Output bits: 96
Status table:

ASCII Port No.: None

Input bits: 0
Output bits: 0

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1-1	CPS-214-00						DC SUMMABLE PS 2
1-2	CPU-x13-0x						CPU 1xMB+
1-3	DDI-353-00		100001	100032			DC IN 24V 4x8
1-4	DDO-353-00				000001	000032	DC OUT 24V 4x8
1-5	AVI-030-00		300001	300009			AN IN 8CH BIPOLAR
1-6	AVO-020-00				400001	400004	AN OUT 4CH VOLT
1-7	CPR-93x-00						RIO Head S908
1-8	...						
1-9	...						
1-10	...						
1-11	...						
1-12	...						
1-13	...						
1-14	...						

Set module parameters

To set parameters for the individual modules, proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	From the **Rack Slot** column select the **1-3** line. **Response:** The **1-3** text box has a dark background, i.e. the **DDI-353-00** module has been selected for editing. **Note:** Parameters are not set for the **CPS-214-00** and **CPU-x13-0x** modules.
2	Select the **Params** command button. **Response:** The **140-DDI-353-00** dialog is opened.
3	Select the **Discrete** option button. **Response:** You return to the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example. **Note:** The modules are sometimes set with different parameters. Help with this can be obtained from the corresponding help texts in the parameter dialog.

Editing Remote Drop

Editing the drop defined second takes place in the dialog **RIO (slot 7) – SY/MAX I/O-St. 2**.

This dialog can be reached in two ways:

- In the **I/O Map** dialog, using the **Edit....** command button, or
- in the **Local Quantum Drop** dialog, using the **Next** command button.

Editing Remote Drop

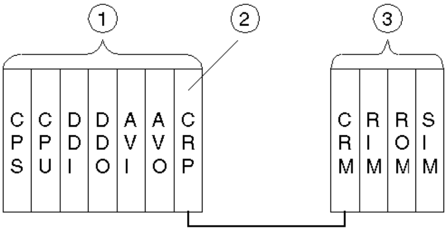
Introduction

This section describes the configuration of the second (remote) drop. The drop has already been defined in Editing the First (local) Drop *(see page 917)*.

To edit the second (remote) drop, the modules must be specified with their I/O references before parameters for the individual modules can be set.

NOTE: To link to the remote network, the coupling module CRM-931-RG must be entered during module mapping.

Quantum – SY/MAX controller



1 Local Quantum drop 1
2 RIO master module
3 SY/MAX drop 2

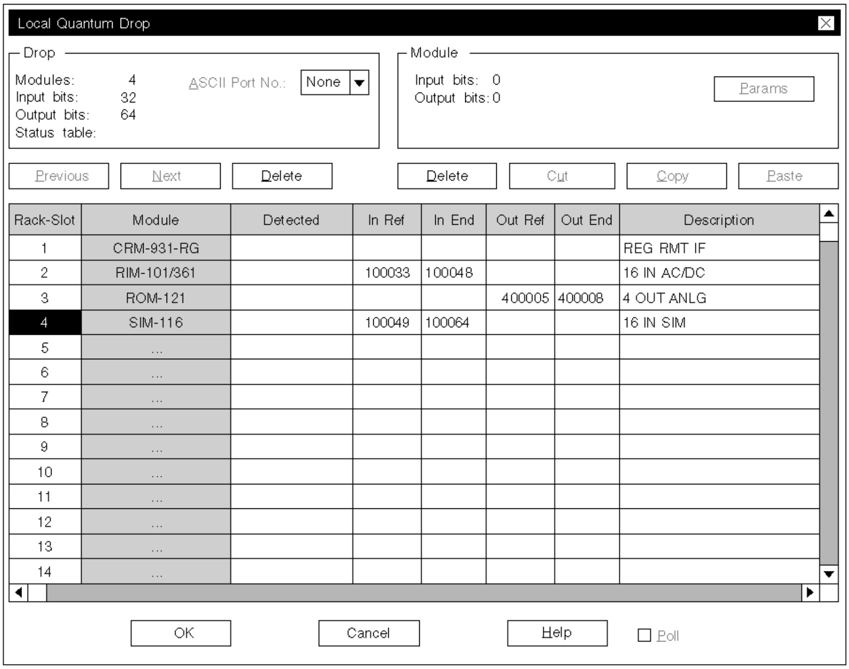
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges proceed as follows in the **RIO (slot 7) – SY/MAX I/O-St. 2** dialog:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column select the **CRM-931-RG** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see dialog representation **RIO (Slot 7) SY/MAX drop 2**).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



Local Quantum Drop

Drop

Modules: 4
Input bits: 32
Output bits: 64
Status table:

ASCII Port No.: None

Module

Input bits: 0
Output bits: 0

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1	CRM-931-RG						REG RMT IF
2	RIM-101/361		100033	100048			16 IN AC/DC
3	ROM-121				400005	400008	4 OUT ANLG
4	SIM-116		100049	100064			16 IN SIM
5	...						
6	...						
7	...						
8	...						
9	...						
10	...						
11	...						
12	...						
13	...						
14	...						

Set module parameters

To parameter the individual modules use the dialog **RIO (slot 7) – SY/MAX I/O-St. 2** and proceed as follows:

Step	Action
1	In the **Slot** column, select line **2**. **Response:** The **2** text box has a dark background, i.e. the **RIM-101/361** module has been selected for editing.
2	Select the **Params** command button. **Response:** The **8030-RIM-101/361** dialog is opened.
3	Select the **Discrete** option button. **Response:** You return to the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example. **Note:** The modules are sometimes set with different parameters. Help with this can be obtained from the corresponding help texts in the parameter dialog.

E.6 Quantum Example - Profibus DP Controller

Overview

This Chapter contains the step-by-step process for the configuration of a Profibus DP controller with the Quantum.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information	926
Profibus DP Export Settings in SyCon	927
Editing Local Drop	929
Importing Profibus DP Configuration	934

General Information

Introduction

Configuring Profibus DP is done using the SyCon (System Configurator) software produced by Hilscher GmbH. It is initially stored there as a file (*.CNF). This generated file is loaded into Concept and is visible in the I/O map of the configurator.

Before the Profibus DP nodes (max. 32) can be imported, a bus controller (CRP 811 00) must be mapped in the drop (Quantum I/O). Depending on the CPU selection in the **Select Extensions** dialog box, a maximum of two to six bus controllers can be inserted.

Profibus DP Export Settings in SyCon

Introduction

SyCon is used to configure Profibus DP. The procedure for this is to be found in the user manual provided by the manufacturer. The settings for the export of the *.CNF file are explained in the following step-by-step instructions.

Preconditions

For CRP-811-00 diagnostics the serial interface of the host computer and the diagnostic interface of the bus controller must be linked with a V24 cable.

To display this diagnostic data, terminal emulation software must be started (e.g. PROCOMM using the settings: 19.2 kBd, 8 data bits, 1 stop bit, no parity).

Defining the Destination Directory

Firstly, specify the destination directory in which all files are to be saved:

Step	Action
1	Select in the main menu **Settings** →**Search Path...**. **Response:** The **Search Path** dialog is opened and is pre-set with the SyCon directory path as the project directory (e.g. C:\HILSCHER GMBH\SYCON\FIELDBUS\PROFIBUS).
2	Enter the path of the Concept directory (e.g. C:\CONCEPT\PROFIBUS) in the **Project Directory** text box. **Note:** You can also accept the default. **Response:** Execution of the **Save** and **Export** menu commands (in the **File** main menu) saves all files in the entered Concept directory.

Generating an Export File

To generate an export file (*.CNF) proceed as follows:

Step	Action
1	Select in the main menu **File** →**Save** →***.PB**. **Response:** The configuration is stored as a database file *.PB in the specified directory.
2	Select in the main menu **File** →**Export** →**ASCII**. **Response:** The configuration is stored as an ASCII file *.CNF in the specified directory.
3	Exit SyCon and start Concept.

Note about Saving

The configuration must always be saved as a database file *.PB first, only then can an ASCII file be generated from the saved *.PB file. Every change must therefore also be saved as a *.PB file first, before an ASCII file can be generated for export.

The files *.PB and *.CNF should always be saved in the same project directory.

Profibus DP Configuration in Concept

After the Profibus DP nodes have been configured in SyCon, the Profibus DP configuration is imported into the Concept I/O map.

An example of configuration and import is described in the chapter "Editing a Local Drop *(see page 929)*".

Editing Local Drop

Introduction

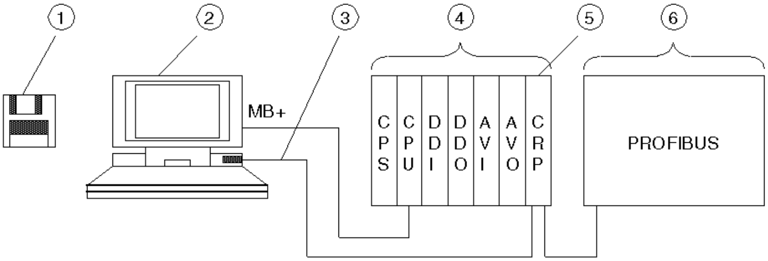
This section describes the configuration of the first (local) drop.

For Profibus DP configuration the CRP-811-00 coupling module must be registered in the I/O map. The configuration defined in SyCon is loaded into Concept as the generated *.CNF file is imported into the parameter dialog of the CRP-811-00 coupling module.

NOTE: For an error free transfer of the Profibus DP configuration, it should be ensured that sufficient memory is available. To optimize storage occupancy open the dialog **PLC Memory Partition** (**PLC Configuration** →**PLC Memory Partition**).

When editing the first (local) drop the modules must be set with their I/O references before the individual modules can be parameterized.

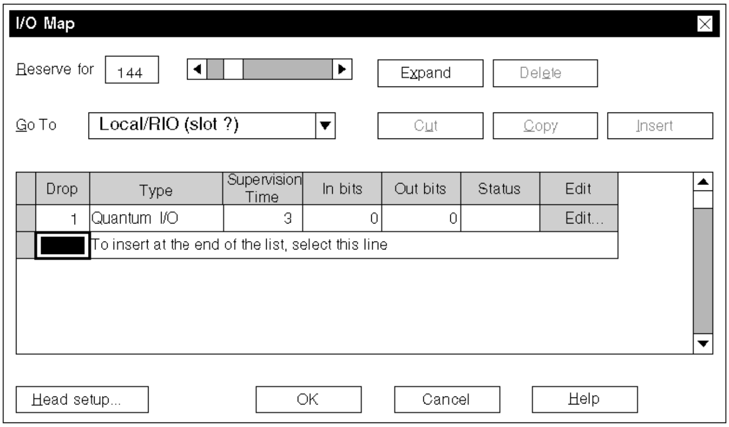
Quantum – Profibus DP controller



1 Device data base for CRP-811-00 (load onto SyCon)
2 Host computer for Concept and SyCon
3 V24 cable
4 Local Quantum drop 1
5 RIO master module
6 Profibus DP configuration (External modules)

Defining Drops

To define drops proceed as follows in the **PLC Configuration** window:

Step	Action
1	Select **PLC Selection**. **Response:** The **PLC Selection** dialog is opened.
2	Select the **Quantum** PLC family and a **CPU x113 xx**. Using **OK** return to the **PLC Configuration** window.
3	Select **Config. Extensions** →**Select Extensions** list. Response: The **Select Extensions** dialog is opened.
4	In the **Profibus DP** list box select the **1** option. **Response:** The coupling module then appears in the **I/O Module Selection** dialog and can be used in the I/O map.
5	Select **I/O Map**. **Response:** The **I/O Map** dialog is opened and the first drop is automatically entered in the table. Dialog Representation  I/O Map Reserve for 144 Go To Local/RIO (slot ?) Drop | Type | Supervision Time | In bits | Out bits | Status | Edit 1 | Quantum I/O | 3 | 0 | 0 | | Edit... To insert at the end of the list, select this line Expand Delete Cut Copy Insert Head setup... OK Cancel Help
6	Select the drop from the **Drop** column. Select the **Edit...** command button. **Response:** You reach the module map.

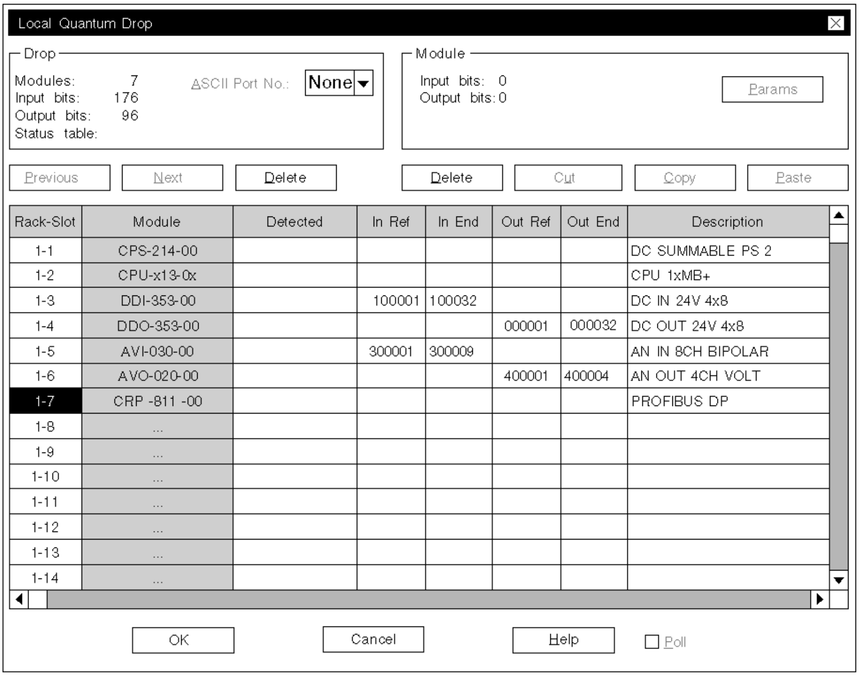
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column, select the **CPS-214-00** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see **Local Quantum Drop** dialog representation).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference of the available address range (**In End** or **Out End** column) is entered automatically.

Dialog Representation

Following module mapping and I/O reference specification, the dialog looks like this:



Drop

Modules: 7
Input bits: 176
Output bits: 96
Status table:

ASCII Port No.: None

Module

Input bits: 0
Output bits: 0

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1-1	CPS-214-00						DC SUMMABLE PS 2
1-2	CPU-x13-0x						CPU 1xMB+
1-3	DDI-353-00		100001	100032			DC IN 24V 4x8
1-4	DDO-353-00				000001	000032	DC OUT 24V 4x8
1-5	AVI-030-00		300001	300009			AN IN 8CH BIPOLAR
1-6	AVO-020-00				400001	400004	AN OUT 4CH VOLT
1-7	CRP -811 -00						PROFIBUS DP
1-8	...						
1-9	...						
1-10	...						
1-11	...						
1-12	...						
1-13	...						
1-14	...						

Parameterization of Modules

To parameterize the individual modules, proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	From the **Rack Slot** column, select line **1-3**. **Response:** The **1-3** text box has a dark background, i.e. the **DDI-353-00** module has been selected for editing. **Note:** The **CPS-214-00** and **CPU-x13-0x** modules are not parameterized.
2	Select the **Params** command button. **Response:** The **140-DDI-353-00** dialog is opened.
3	Select the **Discrete** option button. **Response:** You return to the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example. **Note:** The modules are sometimes parameterized differently. Help with this can be obtained from the corresponding help texts in the parameter dialog.

Importing a Profibus DP Configuration

Importing configured Profibus DP nodes occurs in the parameter dialog of the CRP-811-00 coupling module. This dialog opens when you select the CRP-811-00 row from the I/O map and press the **Params** command button.

Importing Profibus DP Configuration

Introduction

This section describes the import of the Profibus DP configuration. After that, further parameter settings for the master take place and the I/O map can be established.

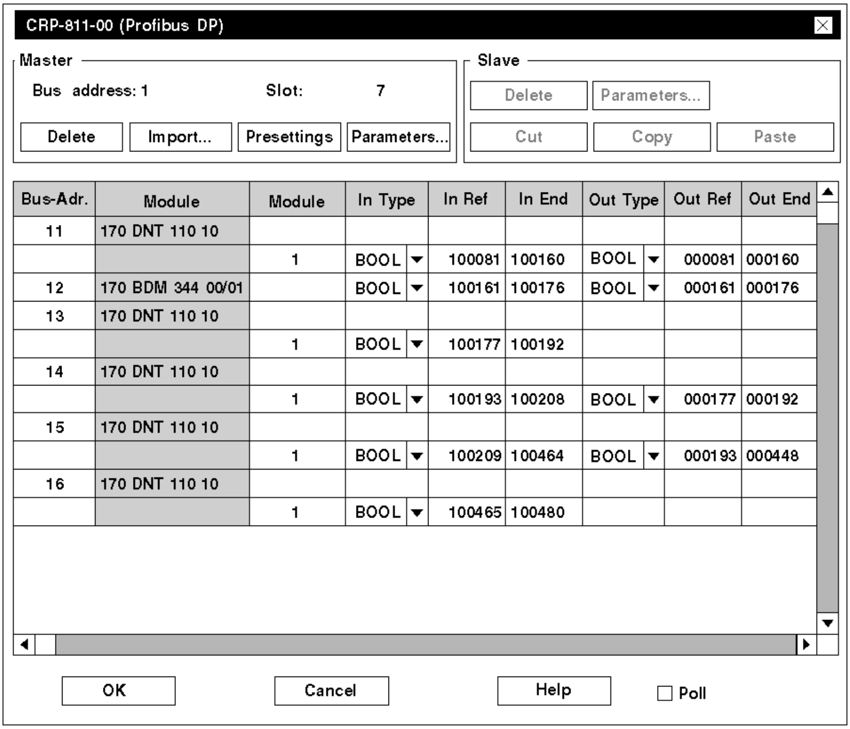
Downloading a Profibus DP Configuration to Concept

To import, proceed as follows:

Step	Action
1	Select the **Import...** command button. **Response:** The **Select Import File** standard window is opened.
2	Enter the path of the previously generated *.CNF file and exit the dialog with **OK**. **Response:** The transfer of the *.CNF file is displayed in the **Import Status** dialog.
3	Close the dialog after the transfer (100%). **Response:** The imported configuration is displayed in the **CRP-811-00 (Profibus DP)** dialog.

Dialog Representation

Following the import of the configuration, the dialog looks like this (view scrolled all the way to the left):

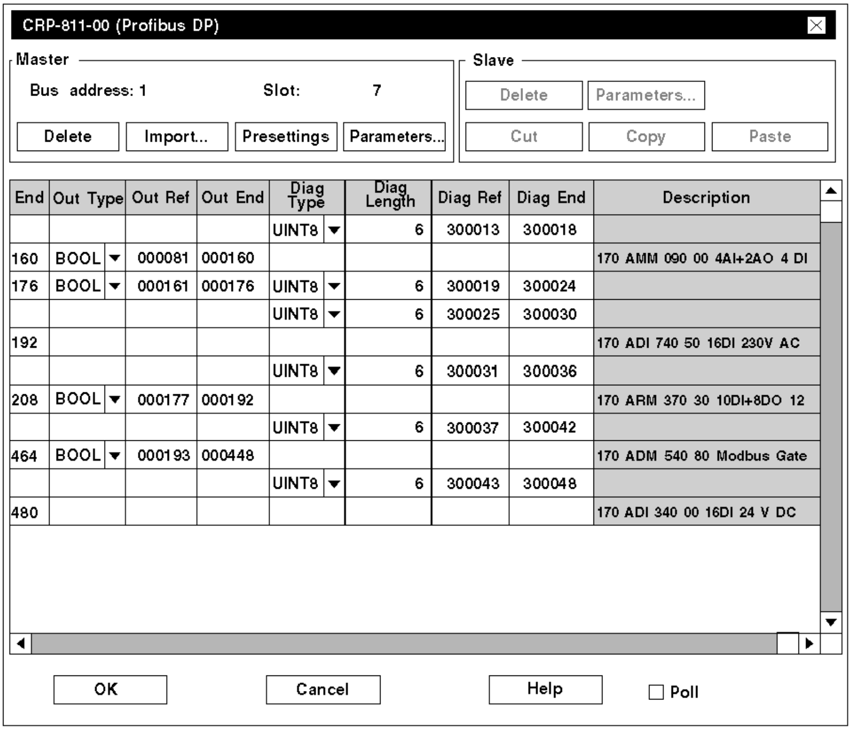


Bus address: 1

Slot: 7

Bus-Adr.	Module	Module	In Type	In Ref	In End	Out Type	Out Ref	Out End
11	170 DNT 110 10							
		1	BOOL	100081	100160	BOOL	000081	000160
12	170 BDM 344 00/01		BOOL	100161	100176	BOOL	000161	000176
13	170 DNT 110 10							
		1	BOOL	100177	100192			
14	170 DNT 110 10							
		1	BOOL	100193	100208	BOOL	000177	000192
15	170 DNT 110 10							
		1	BOOL	100209	100464	BOOL	000193	000448
16	170 DNT 110 10							
		1	BOOL	100465	100480			

Following the import of the configuration, the dialog looks like this (view scrolled all the way to the right):

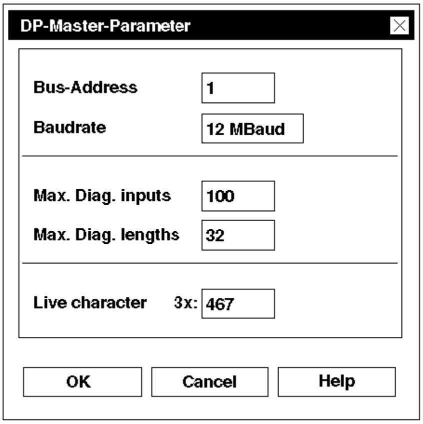


End	Out Type	Out Ref	Out End	Diag Type	Diag Length	Diag Ref	Diag End	Description
				UINT8 ▼	6	300013	300018	
160	BOOL ▼	000081	000160					170 AMM 090 00 4AI+2AO 4 DI
176	BOOL ▼	000161	000176	UINT8 ▼	6	300019	300024	
				UINT8 ▼	6	300025	300030	
192								170 ADI 740 50 16DI 230V AC
				UINT8 ▼	6	300031	300036	
208	BOOL ▼	000177	000192					170 ARM 370 30 10DI+8DO 12
				UINT8 ▼	6	300037	300042	
464	BOOL ▼	000193	000448					170 ADM 540 80 Modbus Gate
				UINT8 ▼	6	300043	300048	
480								170 ADI 340 00 16DI 24 V DC

NOTE: In the **Slave** range, the **Parameter...** command button is used for displaying slave parameters. The slave modules are, however, parameterized in SyCon (see SyCon software user manual).

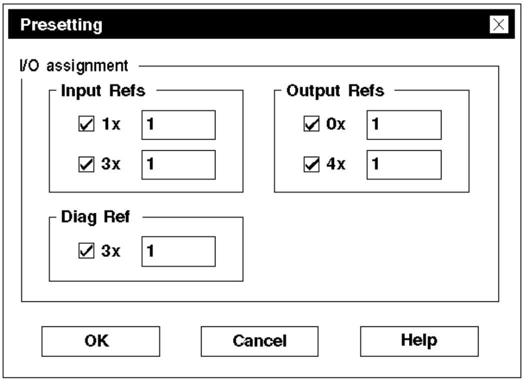
Parameterizing the Master

To parameterize the master, proceed as follows:

Step	Action
1	In the **Master** range, select the **Parameters...** command button. **Response:** The **DP Master Parameters** dialog is opened. Dialog Representation  DP-Master-Parameter Bus-Address: 1 Baudrate: 12 MBaud Max. Diag. inputs: 100 Max. Diag. lengths: 32 Live character 3x: 467
2	Accept the defaults, as shown in the figure above, or redefine them.
3	Close the dialog using **OK**. **Response:** You return to the **CRP-811-00 (Profibus DP)** dialog.

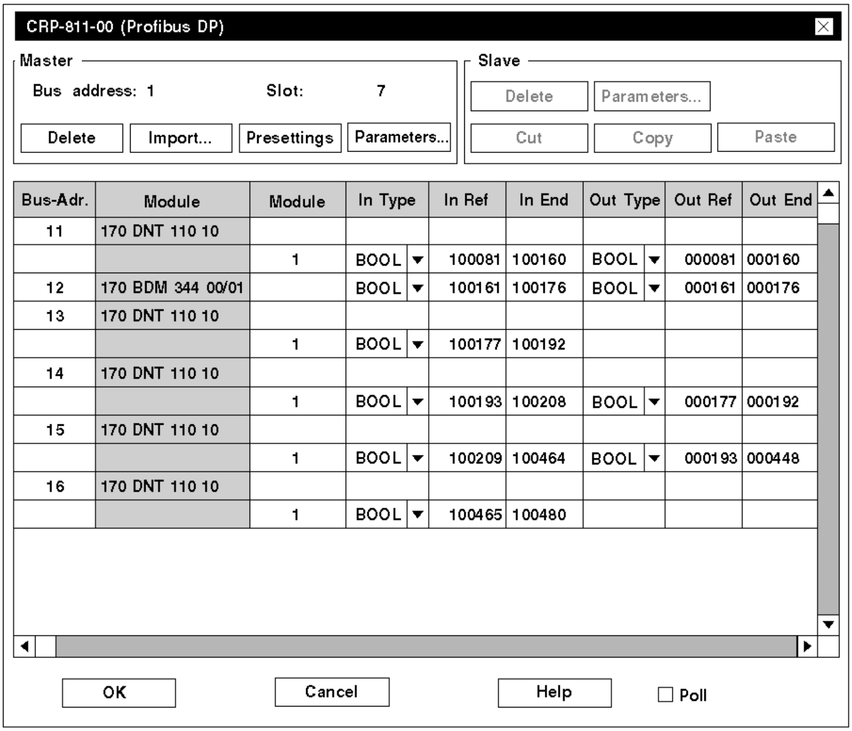
Setting I/O References

To set the I/O references proceed as follows:

Step	Action
1	Select the command button **Preset. ....** The **Preset** dialog is opened. Dialog Representation  Presetting I/O assignment Input Refs ☑ 1x 1 ☑ 3x 1 Output Refs ☑ 0x 1 ☑ 4x 1 Diag Ref ☑ 3x 1
2	Accept the defaults, as shown in the figure above, or redefine them.
3	Close the dialog using **OK**. **Response:** You return to the **CRP-811-00 (Profibus DP)** dialog, in which the defined reference ranges have automatically been entered.

Dialog Representation

After the I/O references have been set the dialog looks like this (view scrolled all the way to the left):



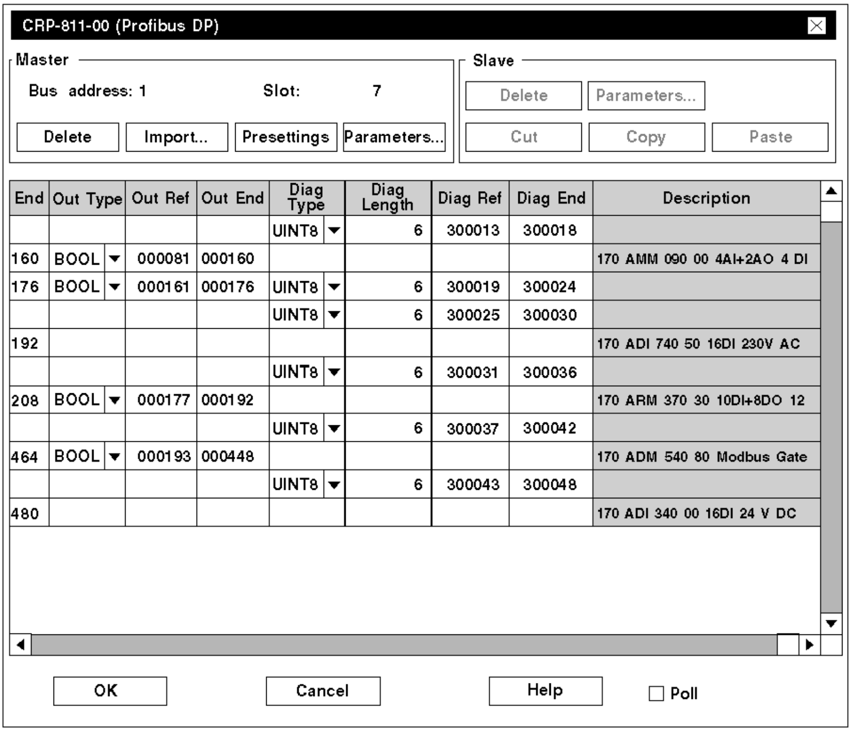
CRP-811-00 (Profibus DP)

Master

Bus address: 1 Slot: 7

Bus-Adr.	Module	Module	In Type	In Ref	In End	Out Type	Out Ref	Out End
11	170 DNT 110 10							
		1	BOOL	100081	100160	BOOL	000081	000160
12	170 BDM 344 00/01		BOOL	100161	100176	BOOL	000161	000176
13	170 DNT 110 10							
		1	BOOL	100177	100192			
14	170 DNT 110 10							
		1	BOOL	100193	100208	BOOL	000177	000192
15	170 DNT 110 10							
		1	BOOL	100209	100464	BOOL	000193	000448
16	170 DNT 110 10							
		1	BOOL	100465	100480			

After the I/O references have been set the dialog looks like this (view scrolled all the way to the right):



CRP-811-00 (Profibus DP)

Master

Bus address: 1 Slot: 7

End	Out Type	Out Ref	Out End	Diag Type	Diag Length	Diag Ref	Diag End	Description
				UINT8	6	300013	300018	
160	BOOL	000081	000160					170 AMM 090 00 4AI+2AO 4 DI
176	BOOL	000161	000176	UINT8	6	300019	300024	
				UINT8	6	300025	300030	
192								170 ADI 740 50 16DI 230V AC
				UINT8	6	300031	300036	
208	BOOL	000177	000192					170 ARM 370 30 10DI+8DO 12
				UINT8	6	300037	300042	
464	BOOL	000193	000448					170 ADM 540 80 Modbus Gate
				UINT8	6	300043	300048	
480								170 ADI 340 00 16DI 24 V DC

□ Poll

E.7 Quantum-Example - Peer Cop

Introduction

In this chapter the configuration of Peer Cop is described step by step.

What's in this Section?

This section contains the following topics:

Topic	Page
Generals to Peer Cop	942
Configuration of Peer Cop	944
Global data transfer	946
Specific data transfer	948

Generals to Peer Cop

Introduction

Peer Cop is a data exchange service provided by the Modbus Plus network. As an overview, imagine that every Modbus Plus network segment (max. 64 nodes) has a global memory, i.e. a certain number of global variables can be read by every node connected to the same segment. The total amount of global variables depends on the number. of connected (and active) nodes, every node can provide up to 32 words (16 bit) to the global memory. Only the 32 words provided by a node can be written by the same node, all other nodes have read only access to these variables. So by definition, there is a maximum of 64 * 32 words of global memory available to a Modbus Plus network segment. Nodes connected to different (through bridges or gateways) segment cannot share global memory.

When a PLC provides 32 words of global memory it does so by assigning holding registers for broadcast, and when the PLC wants to read global variables provided by another Peer Cop node, assigning holding registers to receive them. These registers are called Global Input (from other nodes) and Global Output (what this node provides) get updated cyclically (in case of a PLC after every scan).

To pass Routing Paths

Actually every Modbus Plus node has its own communication processor (the so called Peer processor), in addition to the processor that controls the node specific work (in case of a PLC: solving user logic).

This leads to some routing paths the global data has to pass to traverse from one node to the other:

- From the data provider (e.g. user logic) to the local (most times embedded peer processor).
- From the local peer processor to the other peer processors (this takes the token cycle time of the Modbus Plus network segment, that depends directly on the number of connected nodes).
- From the peer processor of the data receiver to the data receiver itself, (that is usually the user logic in the receiver PLC).

The actual update time depends on the speed of the Modbus Plus network segment and (that’s the big time consumer) the scan times of the data provider and the data receiver.

Send directly

But the sharing of global memory is just the first part of the full Peer Cop service. Since the gobal memory architecture requires a setup (or configuration) for both communication partners, there is another subservice to communicate directly with rather than Configure nodes. This service is somewhat like a master to slave communication, where the master knows what data to send and the slave expects data in a fixed layout and uses this data in a fixed manner (like Terminal I/O). The limit of data that can be sent from the master to the slave is also 32 words. This mode is not global data, since it is sent from one node directly and explicitly to one other node. The sender specifies this as specific output and the receiver as specific input (this specification is hardwired on nonintelligent modules like Terminal I/O). The specific output and input words are also assigned to holding registers when a PLC makes use of this Peer Cop service.

Since both, global and specific data transfer, depend on scan time of the PLC’s which provide and use this data in their logic, there is no big performance difference with the transfer from one holding register to the other registers.

Configuration of Peer Cop

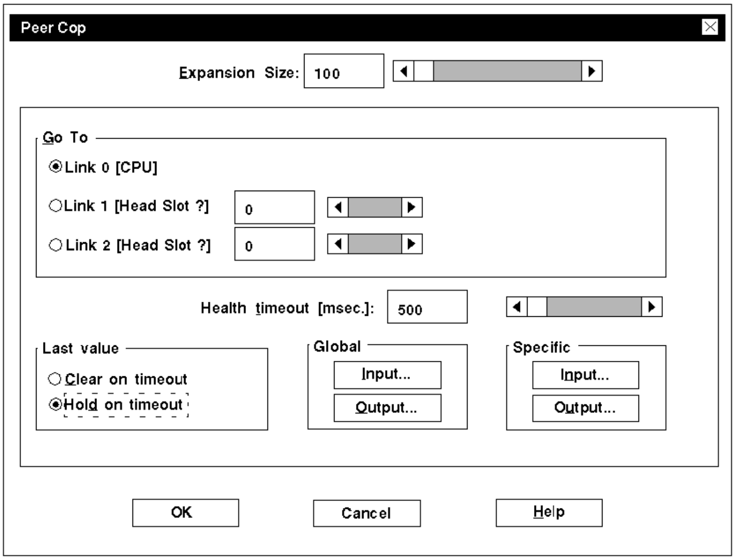
Define Peer Cop functionality

Before configure a Peer Cop you must activate the check box **Peer Cop** in the dialog box **Select Extensions**.

NOTE: Since every PLC can be connected to up to 3 different Modbus Plus network segments, you can setup Peer Cop for every connection separately (remember Peer Cop is reduced to one segment, it doesn't work through bridges).

Peer Cop settings

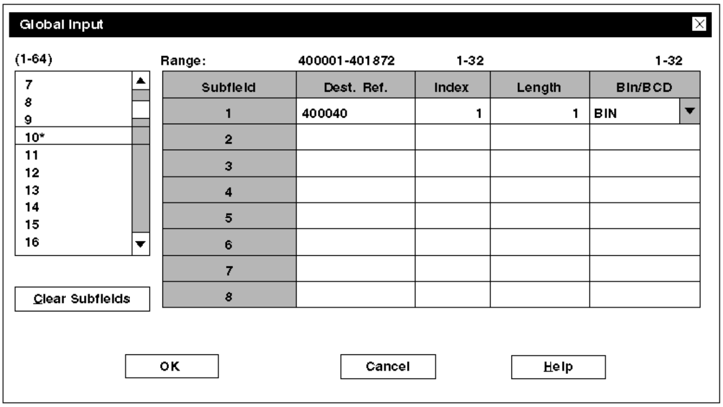
To configure a Peer Cop, proceed with the following steps:

Step	Action
1	In the window **PLC Configuration** with the menu command **Config. Extensions** →**Peer Cop** open the dialog box **Peer Cop**.
2	Assume the default value 100 in the text field **Expansion Size:**. **Note:** This text field is just a space of memory (in words) that gets reserved for future changes (in offline mode) that shall not cause the necessity for a complete download (this is especially importent for direct application setup at a plant).
3	Select the option button **Link 0 (CPU)** in the area **Go To**.
4	Assume the default value 500 in the text field **Health timeout (msec.):**. **Note:** The Health timeout value has the same meaning as it has in the I/O map for local and remote I/O.
5	Select the option button **Hold on timeout** in the area **Last value**. Representation of the dialog:  Peer Cop Expansion Size: 100 Go To ● Link 0 [CPU] ○ Link 1 [Head Slot ?] 0 ○ Link 2 [Head Slot ?] 0 Health timeout [msec.]: 500 Last value ○ Clear on timeout ● Hold on timeout Global: Input... Output... Specific: Input... Output... OK Cancel Help

Global data transfer

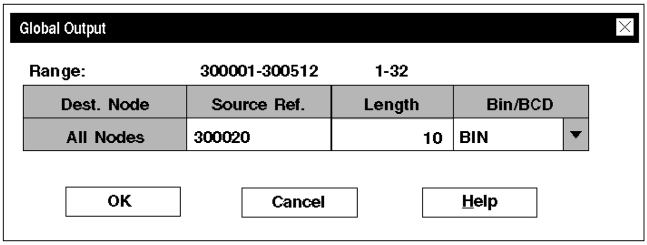
Global Input

For Global Input proceed as follows:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>For global data transfer open the dialog box <b>Gobal Input</b> by clicking the command button <b>Input...</b> in the area <b>Global</b>.</td></tr>
<tr><td>2</td><td>Select node <b>10</b> in the list box of the left side of the dialog box.</td></tr>
<tr><td>3</td><td>Enter the Destination register, the index, the length and the Bin/BCD Code in the text field of the dialog box, as shown in the figure.
Representation of the dialog box:

Global Input
(1-64)
7
8
9
10*
11
12
13
14
15
16
Clear Subfields
OK
Cancel
Help
<table>
<tr><td>Range:</td><td>400001-401872</td><td>1-32</td><td></td><td>1-32</td></tr>
<tr><th>Subfield</th><th>Dest. Ref.</th><th>Index</th><th>Length</th><th>Bin/BCD</th></tr>
<tr><td>1</td><td>400040</td><td>1</td><td>1</td><td>BIN</td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td><td></td></tr>
</table>
<b>Result:</b> The holding register 400040 gets the first word of global output data of node 10, therefore this is global input data for this PLC. If the length value is higher, lets say 2, register 400041 would get the second word of global output data of node 10. The index value declares with what word the assignment shall start, in this case with the first word. The BIN/BCD column gives you the choice of getting the global data formatted either into the usual binary format or into binary coded decimals.
The index value may not be higher than 32, since every node can provide a maximum of 32 word only for global output data. The lenght value may also not be higher than 32 for the same reason.</td></tr>
<tr><td>4</td><td>Close the dialog box <b>Global Input</b> with the command button <b>OK</b>.</td></tr>
</table>

Global Output

For Global Output proceed as follows:

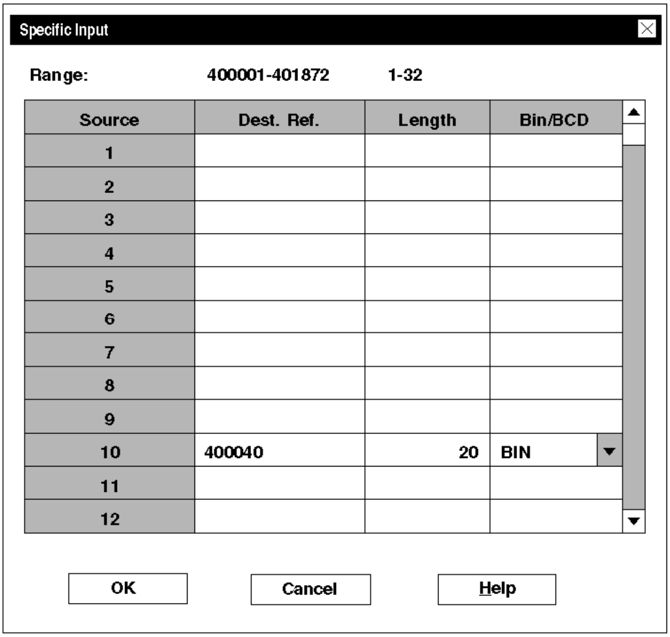
<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Open the dialog box <b>Gobal Output</b> by clicking the command button <b>Output...</b> in the area <b>Global</b>.</td></tr>
<tr><td>2</td><td>Enter the Source register, the length and the Bin/BCD Code in the text field of the dialog box, as shown in the figure.
Representation of the dialog box:

Global Output
Range: 300001-300512 1-32
<table>
<tr><th>Dest. Node</th><th>Source Ref.</th><th>Length</th><th>Bin/BCD</th></tr>
<tr><td>All Nodes</td><td>300020</td><td>10</td><td>BIN</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>Close the dialog box <b>Global Output</b> with the command button <b>OK</b>.</td></tr>
</table>

Specific data transfer

Specific Input

For Specific Input proceed as follows:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>For specific data transfer open the dialog box <b>Specific Input</b> by clicking the command button <b>Input...</b> in the area <b>Specific</b>.</td></tr>
<tr><td>2</td><td>Enter the Destination register, the length and the Bin/BCD Code in the text field of the dialog box, as shown in the figure.
Representation of the dialog box:



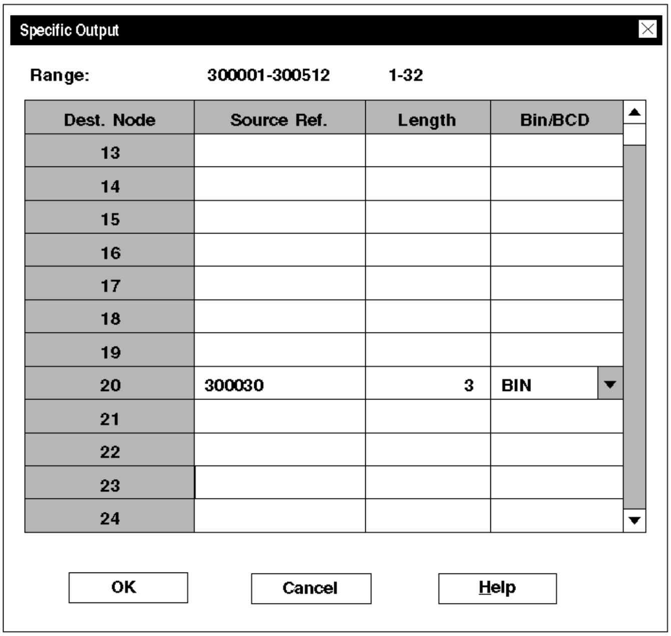
Range: 400001-401872 1-32

<table>
<tr><th>Source</th><th>Dest. Ref.</th><th>Length</th><th>Bin/BCD</th></tr>
<tr><td>1</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td></tr>
<tr><td>9</td><td></td><td></td><td></td></tr>
<tr><td>10</td><td>400040</td><td>20</td><td>BIN</td></tr>
<tr><td>11</td><td></td><td></td><td></td></tr>
<tr><td>12</td><td></td><td></td><td></td></tr>
</table>

<b>Result:</b> If node 10 has declared some specific output, which gets delivered with every token cycle on the Modbus Plus network segment (which is usually faster than the updating by the controller’s user logic), that gets sent to holding register 400040. And if it is more than one word, it gets stored in the following holding register, up to 400019 in this example. The formatting can also be either binary or binary coded decimals.</td></tr>
<tr><td>3</td><td>Close the dialog box with the command button <b>OK</b>.</td></tr>
</table>

Specific Output

For Specific Output proceed as follows:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Open the dialog box <b>Specific Output</b> by clicking the command button <b>Output...</b> in the area <b>Specific</b>.</td></tr>
<tr><td>2</td><td>Enter the Destination Reference register, the length and the Bin/BCD Code option in the text field of the dialog box, as shown in the figure.
Representation of the dialog box:

Specific Output
Range: 300001-300512 1-32
<table>
<tr><th>Dest. Node</th><th>Source Ref.</th><th>Length</th><th>Bin/BCD</th></tr>
<tr><td>13</td><td></td><td></td><td></td></tr>
<tr><td>14</td><td></td><td></td><td></td></tr>
<tr><td>15</td><td></td><td></td><td></td></tr>
<tr><td>16</td><td></td><td></td><td></td></tr>
<tr><td>17</td><td></td><td></td><td></td></tr>
<tr><td>18</td><td></td><td></td><td></td></tr>
<tr><td>19</td><td></td><td></td><td></td></tr>
<tr><td>20</td><td>300030</td><td>3</td><td>BIN</td></tr>
<tr><td>21</td><td></td><td></td><td></td></tr>
<tr><td>22</td><td></td><td></td><td></td></tr>
<tr><td>23</td><td></td><td></td><td></td></tr>
<tr><td>24</td><td></td><td></td><td></td></tr>
</table>

<b>Result:</b> The values or registers 300030 to 300032 will be sent to node 20 (Target Source) in binary format.</td></tr>
<tr><td>3</td><td>Close the dialog box with the command button <b>OK</b>.</td></tr>
</table>

E.8 Compact Example

Editing Local Drop

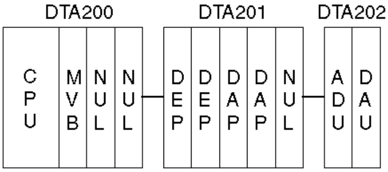
Introduction

This section describes the configuration of the first (local) drop.

When editing the first (local) drop the modules must be set with their I/O references before the individual modules can be parameterized.

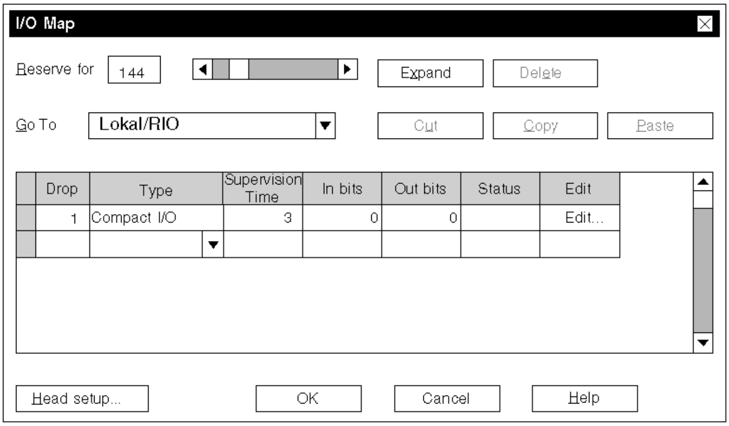
NOTE: The communication module MVB258A is parameterized in the TCN tool (Train Communication Network). A parameterization file (binary file) is generated and imported into the Concept parameter dialog.

Compact controller



Defining Drops

To define drops proceed as follows in the **PLC Configuration** window:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Select <b>PLC Selection</b>.
<b>Response:</b> The <b>PLC Selection</b> dialog is opened.</td></tr>
<tr><td>2</td><td>Select the <b>Compact</b> PLC family and a <b>PC-E984-258</b>. Using <b>OK</b> return to the <b>PLC Configuration</b> window.</td></tr>
<tr><td>3</td><td>Select <b>I/O Map</b>.
<b>Response:</b> The <b>I/O Map</b> dialog is opened and the first drop is automatically entered in the table.
Dialog Representation


Reserve for 144
Go To Lokal/RIO
<table>
<tr><th>Drop</th><th>Type</th><th>Supervision Time</th><th>In bits</th><th>Out bits</th><th>Status</th><th>Edit</th></tr>
<tr><td>1</td><td>Compact I/O</td><td>3</td><td>0</td><td>0</td><td></td><td>Edit...</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>4</td><td>Select the drop from the <b>Drop</b> column.
Select the <b>Edit...</b> command button.
<b>Response:</b> You reach the module map.</td></tr>
</table>

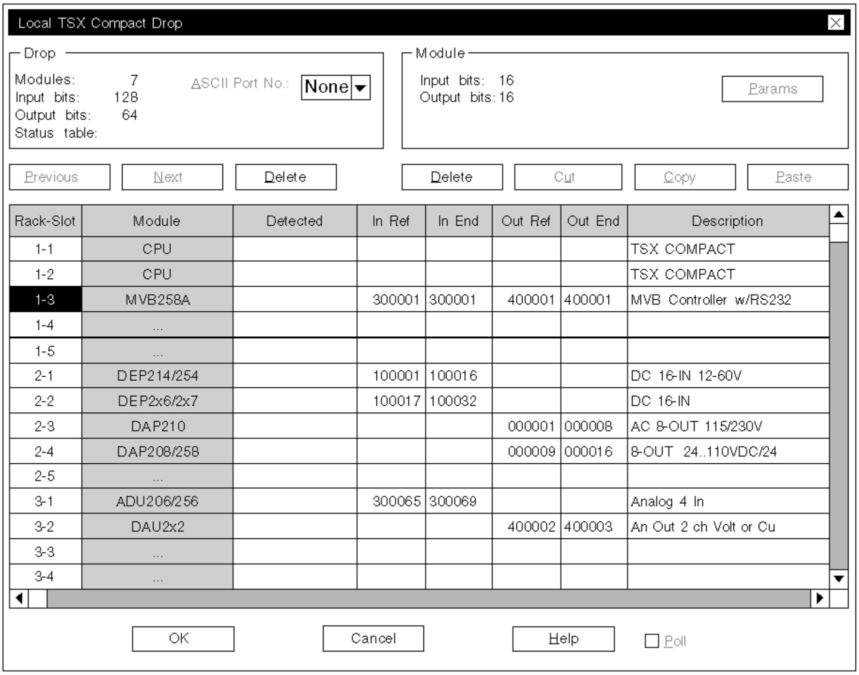
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges proceed as follows in the **Local TSX Compact Drop** dialog:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	In the **Modules** column, select the **MVB258A** module. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see Dialog Representation **Local TSX Compact Drop**).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference of the available address range (**In End** or **Out End** column) is entered automatically.

Dialog Representation

Following module mapping and I/O reference specification, the dialog looks like this:



Drop

Modules: 7
Input bits: 128
Output bits: 64
Status table:

ASCII Port No.: None

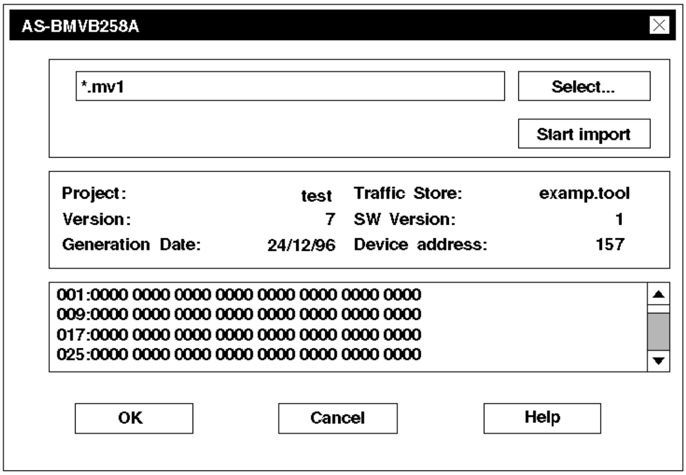
Module

Input bits: 16
Output bits: 16

Rack-Slot	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1-1	CPU						TSX COMPACT
1-2	CPU						TSX COMPACT
1-3	MVB258A		300001	300001	400001	400001	MVB Controller w/RS232
1-4	...						
1-5	...						
2-1	DEP214/254		100001	100016			DC 16-IN 12-60V
2-2	DEP2x6/2x7		100017	100032			DC 16-IN
2-3	DAP210				000001	000008	AC 8-OUT 115/230V
2-4	DAP208/258				000009	000016	8-OUT 24..110VDC/24
2-5	...						
3-1	ADU206/256		300065	300069			Analog 4 In
3-2	DAU2x2				400002	400003	An Out 2 ch Volt or Cu
3-3	...						
3-4	...						

Parameterization of Modules

To parameterize the individual modules proceed as follows in the **Local TSX Compact Drop** dialog:

Step	Action
1	From the **Rack Slot** column, select line **1-3**. **Response:** The **1-3** text box has a dark background, i.e. the **MVB258A** module has been selected for editing. **Note:** The **CPU** module is not parameterized.
2	Select the **Params** command button. **Response:** The **AS-BMVB258A** dialog is opened.
3	Select the **Select** command button. **Response:** The **Select MVB Import File** dialog is opened.
4	Set the path of the parameterization file generated in the TCN tool, and exit the dialog using **OK**. **Response:** The selected parameterization file is displayed in the text box in the **AS-BMVB258A** dialog.
5	Select the **Do Import** command button. **Response:** The project data of the parameterization file is transferred to Concept and displayed in the lower list box. Dialog Representation  AS-BMVB258A Select... Start import OK Cancel Help *.mv1 Project: test Version: 7 Generation Date: 24/12/96 Traffic Store: examp.tool SW Version: 1 Device address: 157 001:0000 0000 0000 0000 0000 0000 0000 0000 009:0000 0000 0000 0000 0000 0000 0000 0000 017:0000 0000 0000 0000 0000 0000 0000 0000 025:0000 0000 0000 0000 0000 0000 0000 0000
6	Exit the dialog using **OK**.
7	Repeat steps 1 to 2 for all the modules in the example. **Note:** The modules are sometimes parameterized differently. Help with this can be obtained from the corresponding help texts in the parameter dialog.

E.9 Atrium Example – INTERBUS Controller

Overview

This Chapter contains the step-by-step process for the configuration of an INTERBUS controller with Atrium (PC based).

What's in this Section?

This section contains the following topics:

Topic	Page
General	956
INTERBUS export settings in CMD	957
Edit local I/O drop	958
Edit remote I/O drop (import INTERBUS configuration)	962

General

Introduction

The configuration of the INTERBUS is done using the PHOENIX software CMD. It is initially stored as a file (*.SVC). This generated file is imported into Concept and is visible in the I/O map of the Configurator.

Before the INTERBUS nodes are imported, set up the first drop (Atrium I/O) with the CPU board (180-CCO-121-01, 180-CCO-241-01 and 180 CCO 241 11) and the INTERBUS master (CRP-660-00/01). A maximum of two INTERBUS masters may be inserted. The diagnostics of the field bus can take place with the CRP-660-0x register in Concept.

INTERBUS export settings in CMD

Introduction

The CMD tool (Configuration Monitoring and Diagnostic tool) is used to configure the INTERBUS. For information about this, refer to the corresponding chapter in the PHOENIX user manual

Preconditions

The serial interface of the host computer and the diagnostic interface of a PC104 board (RS232, to connect to the CMD tool) must be linked with a V24 cable.

Implementing Export Settings

Before you import the configuration into Concept, carry out the following settings in the CMD tool:

Step	Action
1	Select **Configuration** →**Controller Board** →**Type...**.
2	Select **IBS PC104 SC-T**.
3	Deactivate the control button **Automatic Recognition**, select version≤4.40 firmware from the list and confirm your selection with **OK**.
4	Select **File** →**Operating Mode...** **Response:** The **Operating Mode** dialog is opened.
5	Activate the **Configuration (Online)** option button and exit the dialog using **OK**.
6	Select from **Configuration** →**Controller Board** →**Control** the command **Activate Configuration Frame.** Confirm with **Yes.** **Result:** A configuration frame is generated.
7	Select **Configuration** →**Configuration Frame** →**Read Again (from Memory).** **Result:** The configuration is read into the frame.
8	Under **Configuration** →**Parameterization Memory** →**Write ASCII File** select the command **INTERBUS Data (*.SVC)...**. **Response:** The INTERBUS data is stored in a file.
9	Enter the directory and the file name in the open dialog and confirm the entry using **OK**.
10	Select **File** →**Save As...**. **Response:** The INTERBUS project is saved.

Edit local I/O drop

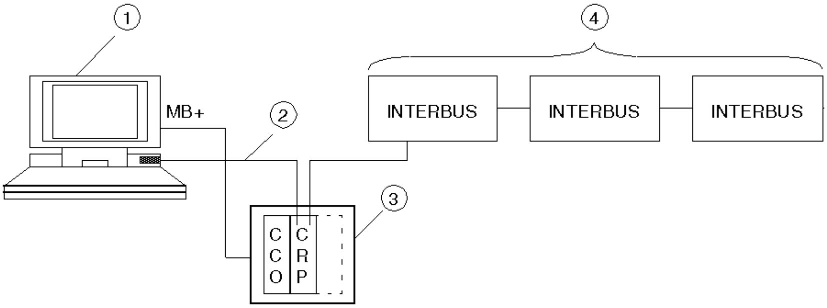
Introduction

In this section the configuration of the first (local) I/O station (drop) is described. The processing sequence begins first with the definition of all I/O drops.

NOTE: To guarantee an error free transfer of the INTERBUS configuration, make sure that sufficient memory is available. To optimise the storage allocation open the **PLC Memory Partition** dialog (**PLC Configuration** →**PLC Memory Partition**).

When editing the first (local) drop the modules must be set with their I/O references before parameters can be set for individual modules.

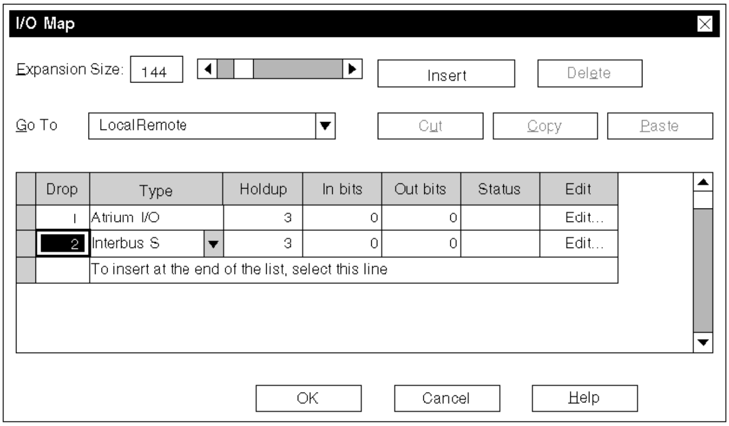
Atrium INTERBUS Controller



1 Programming device for Concept and CMD
2 V24 cable
3 PC104 board on a standard AT platine
4 INTERBUS configuration with the INTERBUS nodes

Define I/O drops

To define the I/O drops, in the **PLC Configuration** window carry out the following steps :

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Select **PLC Selection**.
Result: The **PLC Selection** dialog is opened.</td></tr>
<tr><td>2</td><td>Select the PLC family **Atrium** and under **CPU 180-CCO-241-01**. Clicking **OK** will return you to the **PLC Configuration** window.</td></tr>
<tr><td>3</td><td>Select **I/O Map**.
Result: The **I/O Map** dialog is opened and the first entry in the table is the I/O drop which is automatically entered as **Atrium I/O**.</td></tr>
<tr><td>4</td><td>Select the last row in the table.
Select the command button **Insert**.
Result: In the **Type** column the second I/O drop **Interbus S** is entered.
Dialog display

I/O Map
Expansion Size: 144
Go To LocalRemote
<table>
<tr><th>Drop</th><th>Type</th><th>Holdup</th><th>In bits</th><th>Out bits</th><th>Status</th><th>Edit</th></tr>
<tr><td>1</td><td>Atrium I/O</td><td>3</td><td>0</td><td>0</td><td></td><td>Edit...</td></tr>
<tr><td>2</td><td>Interbus S</td><td>3</td><td>0</td><td>0</td><td></td><td>Edit...</td></tr>
<tr><td></td><td colspan="6">To insert at the end of the list, select this line</td></tr>
</table>
Insert
Delete
Cut
Copy
Paste
OK
Cancel
Help

</td></tr>
<tr><td>5</td><td>Under **Type** you can still select the following for **Interbus S**:
● Interbus S
● Interbus S (PCP)</td></tr>
<tr><td>6</td><td>Select the line 1 (**Atrium I/O**).
Select the command button **Edit**.
Result: The module mapping appears.</td></tr>
</table>

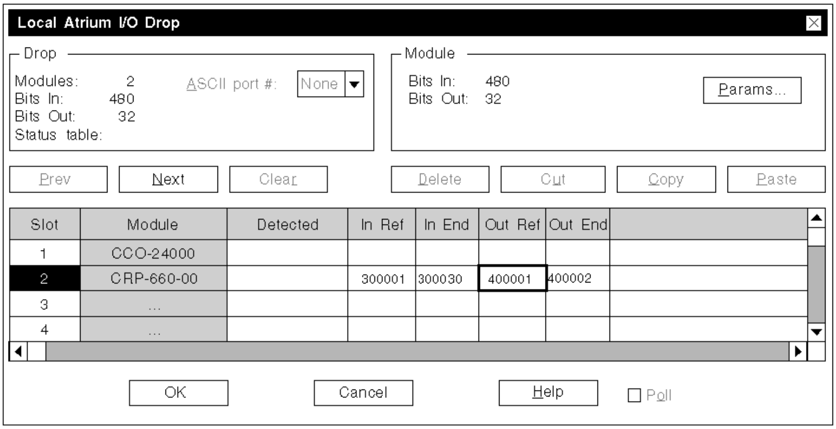
Setting I/O references

In the dialog field **Local Atrium I/O Drop** the INTERBUS Master CRP-660-00 is automatically entered in the I/O map.

To specify the I/O references, in columns **In Ref** and **Out Ref** enter the start references. After the start references have been entered, the end reference for the available address area of the component is shown.

NOTE: Input register references have the prefix 3 (e.g. 300001) and output register references have the prefix 4 (e.g. 400001).

Dialog display



Local Atrium I/O Drop

Drop

Modules: 2
Bits In: 480
Bits Out: 32
Status table:

ASCII port #: None

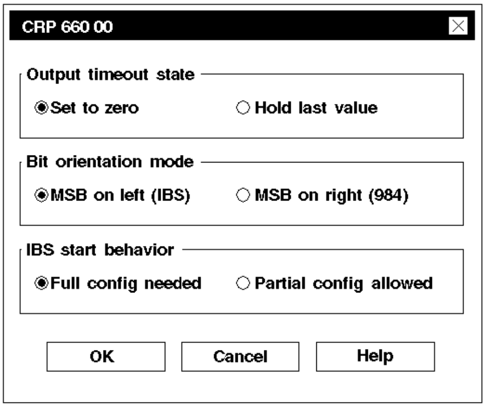
Module

Bits In: 480
Bits Out: 32

Slot	Module	Detected	In Ref	In End	Out Ref	Out End
1	CCO-24000					
2	CRP-660-00		300001	300030	400001	400002
3	...					
4	...					

Set Module Parameters

To set parameters for the INTERBUS master in the **Local Atrium I/O Drop**dialog proceed in the following way:

Step	Action
1	In column **Slot** select the line **2**. **Result:** The text field **2** will then have a dark background, i.e.. the module **CRP-660-00** is selected for editing. **Note:** Parameters are not set for **CCO-24000** module.
2	Select the **Params...** command button. **Result:** The **CRP-660-00** dialog is opened.
3	Press the options button, as shown in the following picture, and exit the dialog by clicking on **OK**. **Note:** Help with setting parameters is obtained via the dialog box's help text. Dialog display  CRP 660 00 Output timeout state: ● Set to zero ○ Hold last value Bit orientation mode: ● MSB on left (IBS) ○ MSB on right (984) IBS start behavior: ● Full config needed ○ Partial config allowed
4	Leave the **Local Atrium I/O Drop** dialog by clicking on **OK**.

Edit remote I/O drop

To edit the remote I/O drop open the dialog box **INTERBUS Drop 2**. This dialog box will take you to the **I/O Map** dialog box, when you click on the command button **Edit...** of the second I/O drop (INTERBUS).

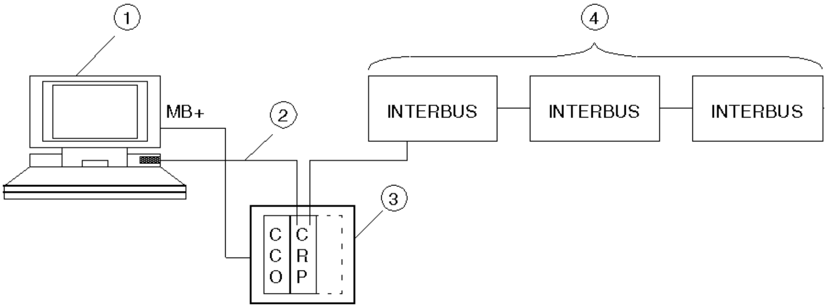
Edit remote I/O drop (import INTERBUS configuration)

Introduction

The INTERBUS configuration import process is described in this section. The map for the I/O reference is in the import dialog, before the transfer of the configuration file is run.

NOTE: The module parameters are set in the CMD tool (see CMD tool user manual), because the imported modules are not recognized in Concept.

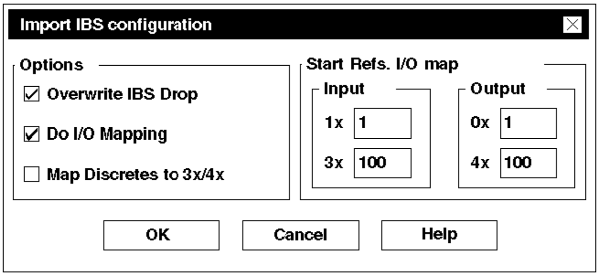
Atrium INTERBUS Controller



1 Programming device for Concept and CMD
2 V24 cable
3 PC104 board on a standard AT platine
4 INTERBUS configuration with the INTERBUS nodes

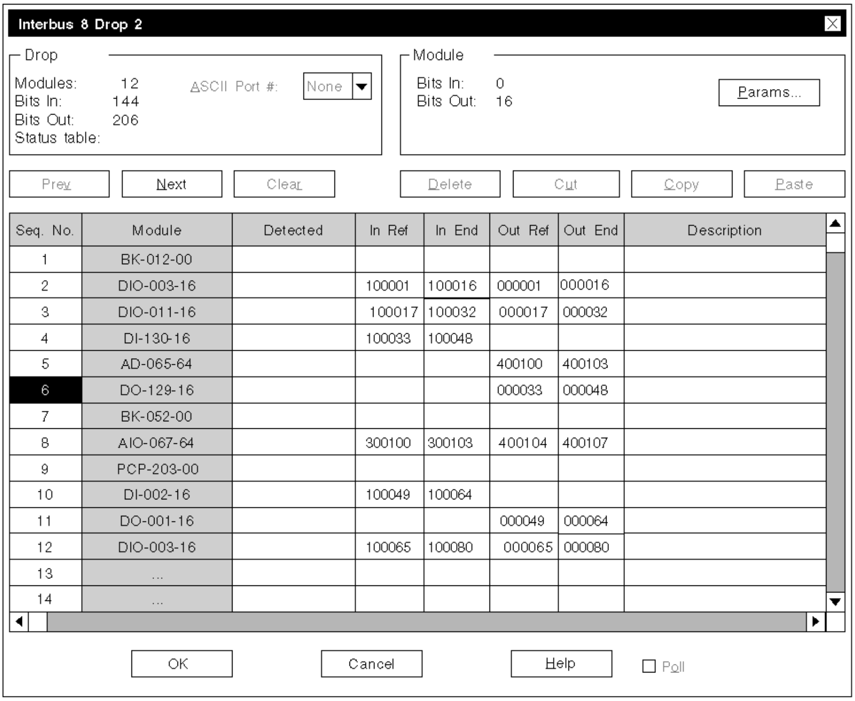
Setting I/O references

To set the address area follow the following steps in the **INTERBUS Drop 2** dialog :

Step	Action
1	Select the command button **Import...**. **Result:** The **Import IBS Configuation** is opened.
2	Activate the checkbox **Overwrite IBS drop**. **Result:** The checkbox **Do I/O mapping** is made available.
3	Activate the **Do I/O mapping** checkbox. **Result:** The **Map Discretes to 3x/4x** area checkbox and the text fields **Input 3x** and **Output 4x** are made available.
4	Deactivate the checkbox **Map Discretes to 3x/4x**. **Result:** The textfields **Input 1x** and **Output 0x** are made available.
5	In the textfields **Input 3x** and **Output 4x** enter the value 100. **Result:** The 3x and 4x address areas of the imported components start with the references 300100 and 400100. **Note:** The 1x- and 0x address areas contain the predefined value 1, i.e. these address area begin with 100001 and 000001. Dialog display  Import IBS configuration Options ☑ Overwrite IBS Drop ☑ Do I/O Mapping □ Map Discretes to 3x/4x Start Refs. I/O map Input 1x 1 3x 100 Output 0x 1 4x 100
6	You can exit the dialog with **OK**. **Result:** The dialog **Select Import File** is opened.
7	Enter the path in the *.SVC configuration file. Select **OK**. **Result:** The dialog **Import Status** is opened, the file transfer starts and the import status is shown.
8	After the transfer (100%) close the dialog. **Result:** The imported INTERBUS configuration is shown in the **INTERBUS Drop 2** in the I/O map.

Dialog display

After the INTERBUS configuration the dialog looks, for example, as follows:



Interbus 8 Drop 2

Drop

Modules: 12
Bits In: 144
Bits Out: 206
Status table:

ASCII Port #: None

Module

Bits In: 0
Bits Out: 16

Seq. No.	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1	BK-012-00						
2	DIO-003-16		100001	100016	000001	000016	
3	DIO-011-16		100017	100032	000017	000032	
4	DI-130-16		100033	100048			
5	AD-065-64				400100	400103	
6	DO-129-16				000033	000048	
7	BK-052-00						
8	AIO-067-64		300100	300103	400104	400107	
9	PCP-203-00						
10	DI-002-16		100049	100064			
11	DO-001-16				000049	000064	
12	DIO-003-16		100065	100080	000065	000080	
13	...						
14	...						

E.10 Momentum Example - Remote I/O Bus

Overview

This Chapter contains the step-by-step process for the configuration of a remote I/O bus (Momentum).

What's in this Section?

This section contains the following topics:

Topic	Page
General Information	966
Editing local drop	967
Example 10 – Editing Remote Drops (I/O Bus)	971

General Information

Introduction

TSX Momentum is a modular system. Bus adapters (e.g. 170 INT 110 00) and CPU adapters (e.g. 171-CCC-760-10-IEC) work in conjunction with an I/O unit as independent modules. In order to function properly, each I/O unit must be equipped with an adapter.

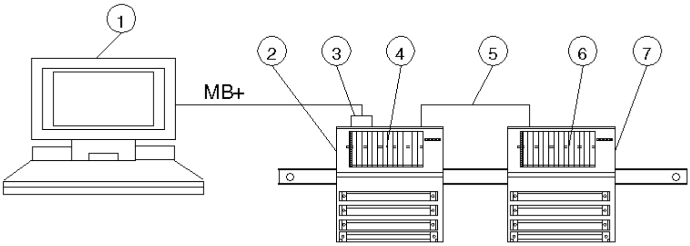
Editing local drop

Introduction

This section describes the configuration of the first (local) drop. The processing sequence begins first of all with the definition of all drops.

When editing the first (local) drop the modules must be set with their I/O references before parameters can be set forindividual modules.

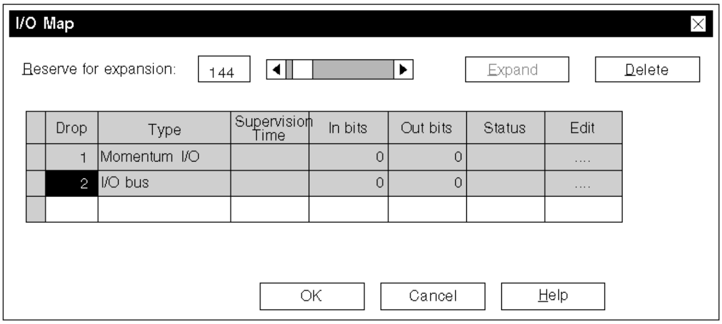
Momentum – remote controller with I/O bus



1 Host Computer
2 I/O unit e.g. 170-AAI-030-00
3 Interface adapter
4 CPU adapter e.g. 171-CCC-760-10-IEC
5 I/O bus interface e.g. 172-PNN-210-22
6 Bus adapter e.g. 170-INT-110-00
7 I/O unit e.g. 170-AMM-090-00

Defining Drops

To define drops proceed as follows in the **PLC Configuration** window:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Select <b>PLC Selection</b>.
<b>Response:</b> The <b>PLC selection</b> dialog is opened.</td></tr>
<tr><td>2</td><td>Select the PLC family <b>Momentum</b> and CPU <b>171-CCC-760-10-IEC</b>. Use <b>OK</b> return to the <b>PLC Configuration</b> window.</td></tr>
<tr><td>3</td><td>Select <b>I/O Map</b>.
<b>Response:</b> The <b>I/O Map</b> dialog is opened and the first drop is automatically entered in the table.</td></tr>
<tr><td>4</td><td>Select the last line in the table.
Select the <b>Insert</b> command button.
<b>Response:</b> The second drop is entered in the <b>Type</b> column.
<b>Note:</b> Only one I/O bus can be configured.
Dialog display


Reserve for expansion: 144
<table>
<tr><th>Drop</th><th>Type</th><th>Supervision Time</th><th>In bits</th><th>Out bits</th><th>Status</th><th>Edit</th></tr>
<tr><td>1</td><td>Momentum I/O</td><td></td><td>0</td><td>0</td><td></td><td>....</td></tr>
<tr><td>2</td><td>I/O bus</td><td></td><td>0</td><td>0</td><td></td><td>....</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>5</td><td>Select the drop from the <b>Drop</b> column.
Select the <b>Edit...</b> command button.
<b>Response:</b> You reach the module map.</td></tr>
</table>

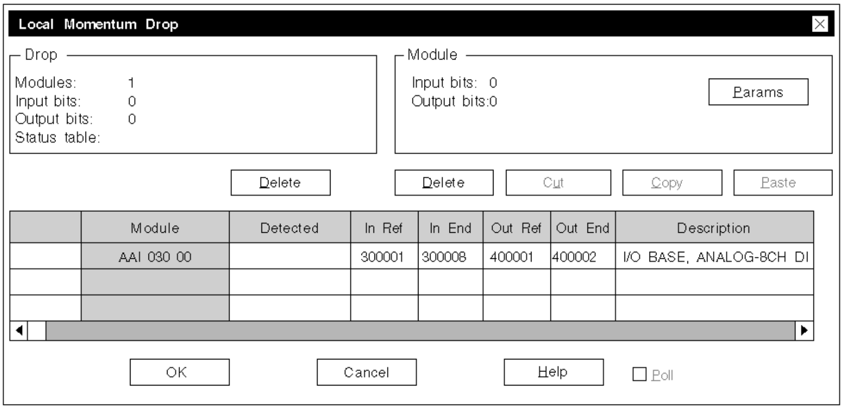
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	Select from the column **Modules**, the module**AAI-030-00**. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see **Local Momentum drop**).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



Local Momentum Drop

Drop

Modules: 1
Input bits: 0
Output bits: 0
Status table:

Module

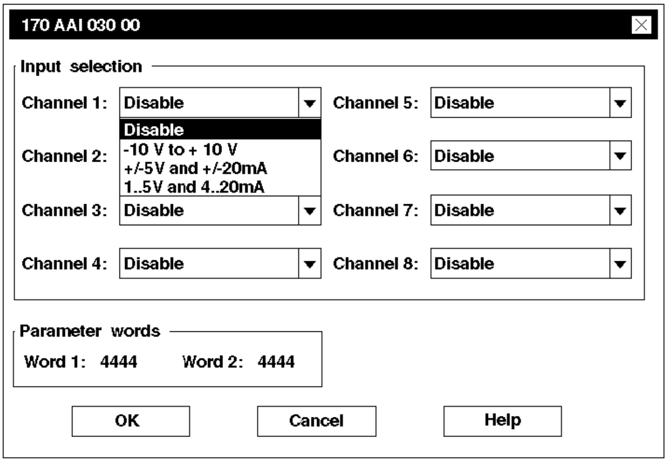
Input bits: 0
Output bits:0

	Module	Detected	In Ref	In End	Out Ref	Out End	Description
	AAI 030 00		300001	300008	400001	400002	I/O BASE, ANALOG-8CH DI

NOTE: With this addressing the 8 measurements of the AAI-030-00 are to be found in the words 300001-300008. The parameters are in the words 400001 and 400002.

Set Module Parameters

To set parameters for the module proceed as follows in the **Local Momentum Drop** dialog:

Step	Action
1	Select the **Params** command button. **Response:** The **170-DDI-353-00** dialog is opened.
2	Select the signal conditions for the input and output channels from the list boxes and exit the dialog using **OK**. **Note:** Help with this can be obtained from the corresponding help text in the parameter dialog. **Response:** The parameter settings are automatically allocated to the addresses 400001 and 400002. Dialog display  170 AAI 030 00 Input selection Channel 1: Disable Disable -10 V to + 10 V +/-5V and +/-20mA 1..5V and 4..20mA Channel 2: Channel 3: Disable Channel 4: Disable Channel 5: Disable Channel 6: Disable Channel 7: Disable Channel 8: Disable Parameter words Word 1: 4444 Word 2: 4444
3	Exit the dialog using **OK**. **Response:** You return automatically to the **I/O Map** dialog.

Editing Remote Drops (I/O bus)

To edit the remote drop open the **RIO I/O Bus Drop** dialog. This dialog is reached via the **I/O Map** dialog by pressing the **Edit...** command button in the second drop (I/O bus).

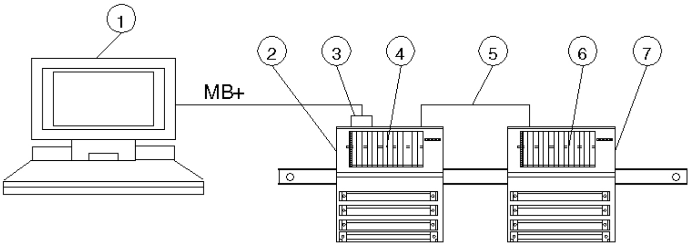
Example 10 – Editing Remote Drops (I/O Bus)

Introduction

This section describes the configuration of the Momentum I/O bus. The drop has already been defined in Editing the First (local) Drop *(see page 967)*.

When editing the I/O bus the modules must be specified with their I/O references before the individuals modules can be parameterized.

Momentum – remote controller with I/O bus



1 Host Computer
2 I/O unit e.g. 170-AAI-030-00
3 Interface adapter e.g. 172-PNN-210-22
4 CPU adapter e.g. 171-CCC-760-10-984
5 I/O bus interface
6 Bus adapter e.g. 170-INT-110-00
7 I/O unit e.g. 170-AMM-090-00

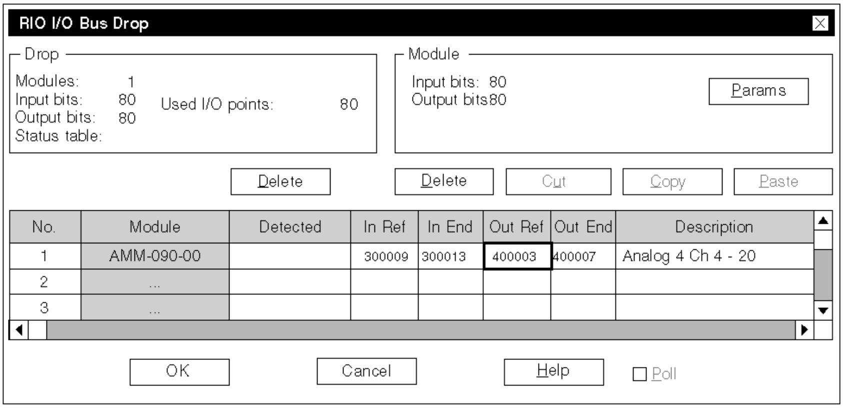
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges proceed as follows in the **Local TSX Compact Drop** dialog:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	Select from the column **Modules**, the module**AMM-090-00**. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	Repeat steps 1 to 3 for all the modules in the example (see **Local Quantum Drop** dialog representation).
5	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



RIO I/O Bus Drop

Drop

Modules: 1
Input bits: 80
Output bits: 80
Status table:
Used I/O points: 80

Module

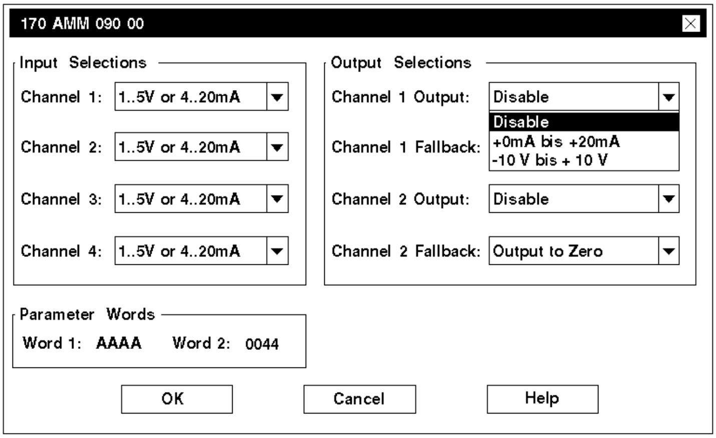
Input bits: 80
Output bits80

No.	Module	Detected	In Ref	In End	Out Ref	Out End	Description
1	AMM-090-00		300009	300013	400003	400007	Analog 4 Ch 4 - 20
2	...						
3	...						

NOTE: With this addressing, the 4 measurements of the AMM-090-00 are to be found in the words 300009-300013. The parameters are in the words 400009-400013.

Set Module Parameters

To set parameters for the module proceed as follows in the **RIO I/O bus drop** dialog:

Step	Action
1	Select column **No.** line **1**. **Response:** The **1** text box has a dark background, i.e. the **AMM-090-00** module has been selected for editing.
2	Select the **Params** command button. **Response:** The **170-AMM-090-00** dialog is opened.
3	Select the signal states for the input and output channels from the list boxes and exit the dialog using **OK**. **Note:** Help with this can be obtained from the help text in the parameter dialog. **Response:** The parameter settings are automatically allocated to the addresses 400009-400013. Dialog display  170 AMM 090 00 Input Selections Channel 1: 1..5V or 4..20mA Channel 2: 1..5V or 4..20mA Channel 3: 1..5V or 4..20mA Channel 4: 1..5V or 4..20mA Output Selections Channel 1 Output: Disable Channel 1 Fallback: Channel 2 Output: Disable Channel 2 Fallback: Output to Zero Parameter Words Word 1: AAAA Word 2: 0044 Disable +0mA bis +20mA -10 V bis + 10 V OK Cancel Help

E.11 Momentum Example - Ethernet Bus System

Overview

This chapter contains step-by-step instructions for the configuration of an Ethernet bus system with Momentum.

What's in this Section?

This section contains the following topics:

Topic	Page
Configure Ethernet	975
Network Configuration in Different Operating Systems	976
Editing local drop	986
Create online connection	990

Configure Ethernet

Preconditions

To configure an Ethernet bus system, the following preconditions must be fulfilled:

- PCI network cards in the host computer
- Installation of the network card driver
- Setting Ethernet interface parameters
- Addressing the M1 Ethernet CPU

Installing the PCI network card

For a link to an Ethernet bus system an Ethernet interface located on a PCI network card must be available in the host computer. This card can be upgraded in PCs, as long as a PCI slot is available. Information about this can be found in the computer manufacturer's user manual.

Network configuration

Network configurations for different operating systems are given in section *Network Configuration in Different Operating Systems, page 976*.

Installing Drivers

Following the installation of the PCI network card the drivers, which come with the network card, must be installed.

To proceed further, the IP address of the network card is required (it may be necessary to contact network administrator).

Addressing the M1 Ethernet CPU

The M1 Ethernet CPU does not have an IP address when supplied, and must therefore be determined in the **Ethernet / I/O Scanner** dialog. The address for the gateway and Subnet Mask is also determined in this dialog.

The IP address can be assigned via the system administrator or the BOOTP server.

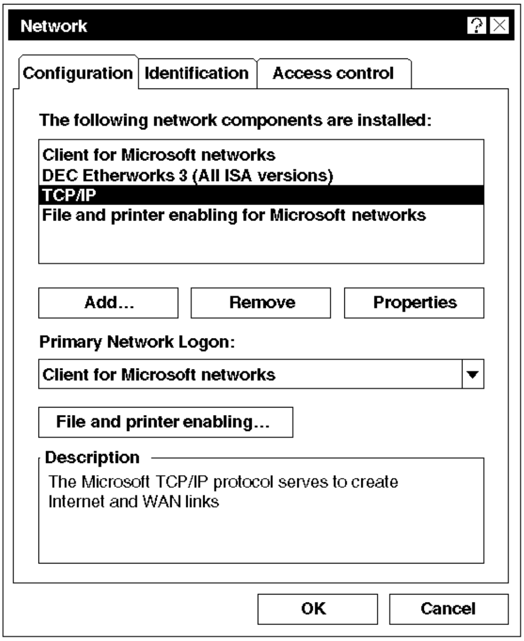
NOTE: It is important to ensure that the IP address has not already been assigned to another device. Double addressing causes an unforeseeable function in the network.

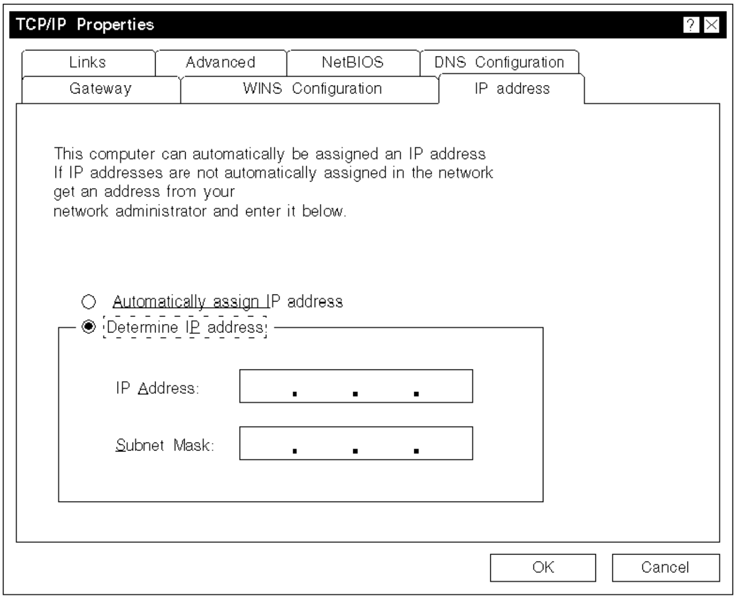
After addressing, saving to Flash is recommended (**Online Control Panel →Flash Program...**), so that the settings are not lost in case of a power outage.

Network Configuration in Different Operating Systems

Network configuration in Win 98

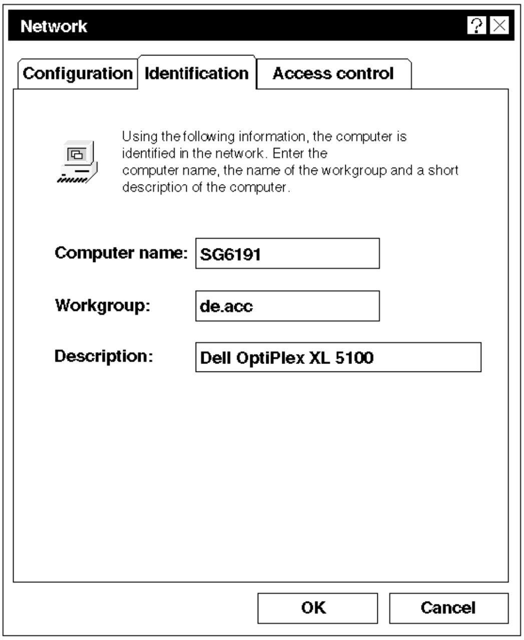
Declare this IP address in the operating system as follows:

Step	Action
1	Select **Start** →**Settings** →**Control Panel** →**Network**. **Response:** The **Network** dialog box is opened. Dialog display  The following network components are installed: Client for Microsoft networks DEC Etherworks 3 (All ISA versions) TCP/IP File and printer enabling for Microsoft networks Primary Network Logon: Client for Microsoft networks Description: The Microsoft TCP/IP protocol serves to create Internet and WAN links Network Configuration Identification Access control Add... Remove Properties File and printer enabling... OK Cancel
2	Select the register **Configuration**. Select the network connection **TCP/IP**.

Step	Action
3	Select the **Properties** command button. **Response:** The **TCP/IP Properties** dialog is opened. Dialog display  TCP/IP Properties Links Advanced NetBIOS DNS Configuration Gateway WINS Configuration IP address OK Cancel This computer can automatically be assigned an IP address If IP addresses are not automatically assigned in the network get an address from your network administrator and enter it below. ○ Automatically assign IP address ● Determine IP address IP Address: . . . Subnet Mask: . . .
4	Select the register **IP Address** and make the following settings. **Response:** The programming device is then registered for network operation with the IP address.

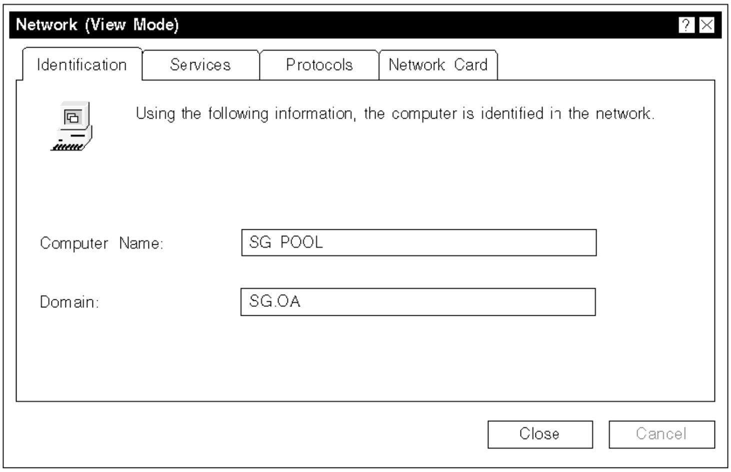
Computer Identification in Win 98/NT

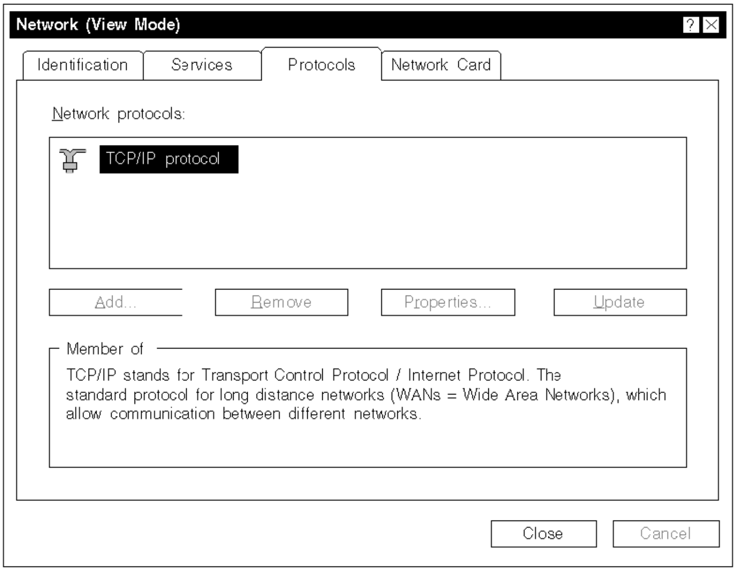
The information is used to identify the computer in the network:

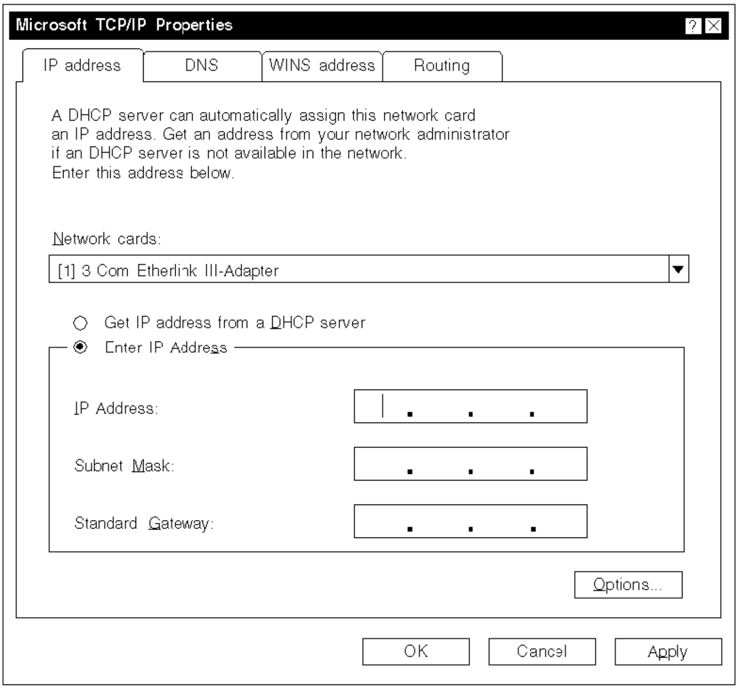
Step	Action
1	Select **Start** →**Settings** →**Control Panel** →**Network**. **Response:** The **Network** dialog box is opened.
2	Select the register **Identification**. Enter the computer name, the name of the workgroup and a short description of the computer. Dialog display  Network Configuration Identification Access control OK Cancel Using the following information, the computer is identified in the network. Enter the computer name, the name of the workgroup and a short description of the computer. Computer name: SG6191 Workgroup: de.acc Description: Dell OptiPlex XL 5100
3	Exit the dialog using **OK**.

Network configuration in Win NT

Declare this IP address in the operating system as follows:

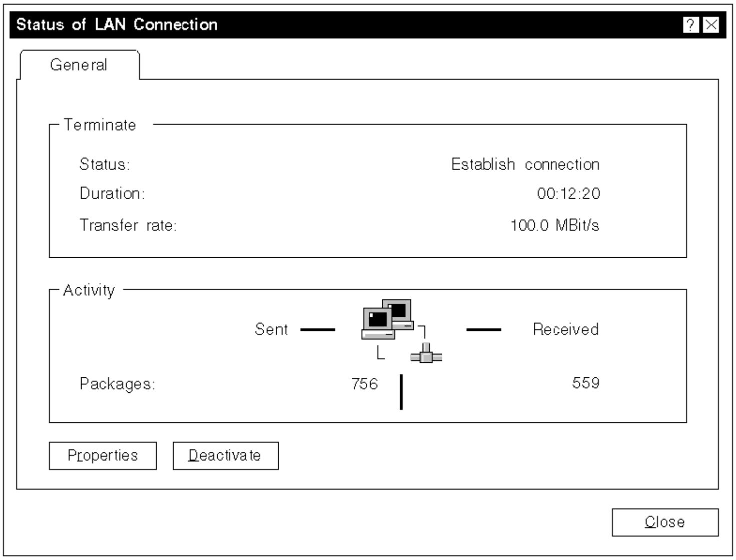
Step	Action
1	Select **Start** →**Settings** →**Control Panel** →**Network**. **Response:** The **Network** dialog box is opened. Dialog display  Network (View Mode) Identification Services Protocols Network Card Close Cancel Using the following information, the computer is identified in the network. Computer Name: SG POOL Domain: SG.OA

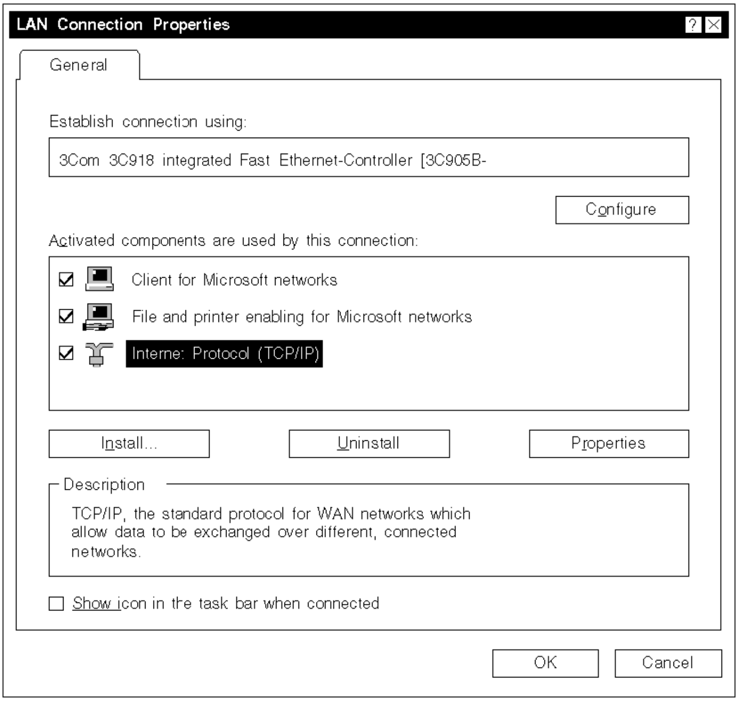
Step	Action
2	Select the register **Protocols**. Dialog display  TCP/IP stands for Transport Control Protocol / Internet Protocol. The standard protocol for long distance networks (WANs = Wide Area Networks), which allow communication between different networks. Select the network connection **TCP/IP Protocol**.

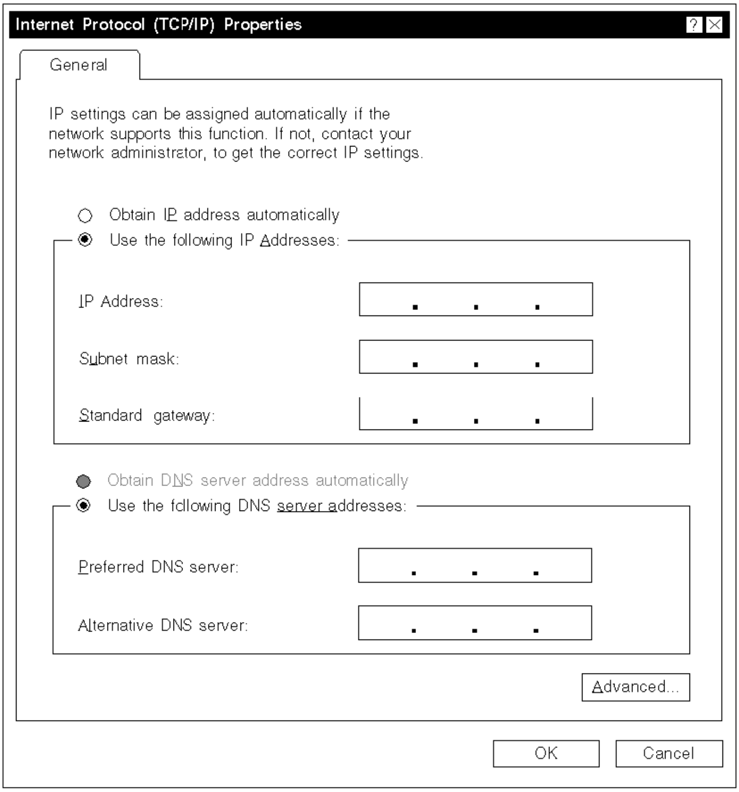
Step	Action
3	Select the **Properties** command button. **Response:** The **Microsoft TCP/IP Properties** dialog box is opened. Dialog display  Microsoft TCP/IP Properties A DHCP server can automatically assign this network card an IP address. Get an address from your network administrator if an DHCP server is not available in the network. Enter this address below. Network cards: [1] 3 Com Etherlink III-Adapter ○ Get IP address from a DHCP server ● Enter IP Address IP Address: . . . Subnet Mask: . . . Standard Gateway: . . . IP address DNS WINS address Routing Options... OK Cancel Apply
4	Select the register **IP Address** and make the following settings. **Response:** The programming device is then registered for network operation with the IP address.

Network configuration in Win 2000

Declare this IP address in the operating system as follows:

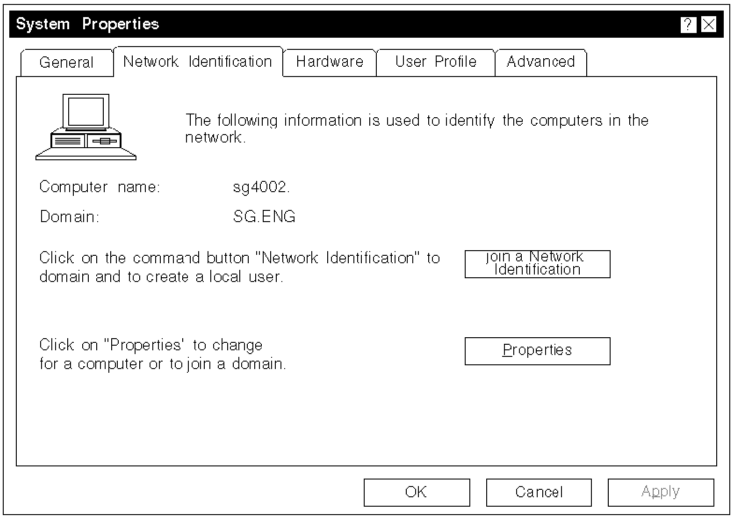
Step	Action
1	Select **Start** →**Settings** →**Network and Dial-Up Connections**. **Response:** The **Network and Dial-Up Connections** window is opened.
2	Select the **LAN Connection** icon. **Response:** The **LAN Connection status** dialog box is opened. Dialog display  Status of LAN Connection Terminate Status: Establish connection Duration: 00:12:20 Transfer rate: 100.0 MBit/s Activity Sent — Received Packages: 756 559

Step	Action
3	Select the **Properties** command button. **Response:** The **LAN Connection Properties** dialog box is opened. Dialog display  LAN Connection Properties General Establish connection using: 3Com 3C918 integrated Fast Ethernet-Controller [3C905B- Activated components are used by this connection: ☑ Client for Microsoft networks ☑ File and printer enabling for Microsoft networks ☑ Interne: Protocol (TCP/IP) Description TCP/IP, the standard protocol for WAN networks which allow data to be exchanged over different, connected networks. ☐ Show icon in the task bar when connected ? Configure Install... Uninstall Properties OK Cancel
4	Select the network connection **Internet Protocol (TCP/IP)**.

Step	Action
5	Select the **Properties** command button. **Response:** The **Internet Protocol (TCP/IP) Properties** dialog box is opened. Dialog display  Internet Protocol (TCP/IP) Properties General IP settings can be assigned automatically if the network supports this function. If not, contact your network administrator, to get the correct IP settings. ○ Obtain IP address automatically ● Use the following IP Addresses: IP Address: . . . Subnet mask: . . . Standard gateway: . . . ● Obtain DNS server address automatically ● Use the following DNS server addresses: Preferred DNS server: . . . Alternative DNS server: . . . Advanced... OK Cancel
6	Make the settings there. **Response:** The programming device is then registered for network operation with the IP address.

Computer Identification in Win 2000

The information is used to identify the computer in the network:

Step	Action
1	Select **Start** →**Settings** →**Control Panel** →**System**. **Response:** The **System Properties** window is opened.
2	Select the register **Network Identification**. Dialog display  System Properties General Network Identification Hardware User Profile Advanced join a Network Identification Properties OK Cancel Apply The following information is used to identify the computers in the network. Computer name: sg4002. Domain: SG.ENG Click on the command button "Network Identification" to domain and to create a local user. Click on "Properties' to change for a computer or to join a domain.
3	Select the **Network ID** command button. **Response:** The assistant for creating a user on the network is started. Or select the **Properties** command button. **Response:** The **Identification Changes** dialog box is opened.
4	Exit the dialog using **OK**.

Editing local drop

Introduction

This section describes the configuration of the local I/O station (Drop). The processing sequence begins first of all with the definition of the drop.

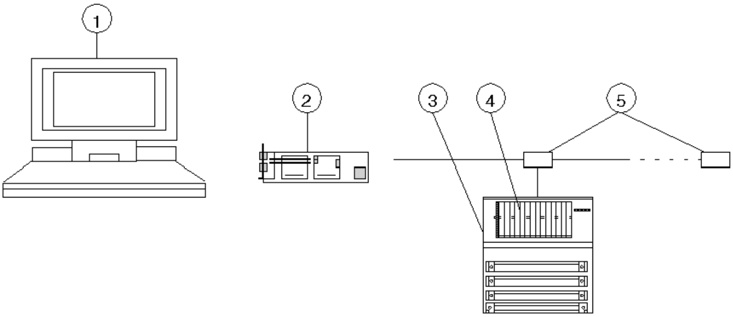
When editing the local I/O station (Drop) the I/O unit must be specified with its I/O references before parametering of the individual assemblies can take place.

NOTE: Only particular CPUs can be used for the Ethernet bus configuration.

The following CPUs are available:

- 171 CCC 980 30
- 171 CCC 960 30
- 171 CCC 980 20
- 171 CCC 960 20

Momentum - Ethernet Bus System



1 Host Computer
2 Ethernet network card
3 I/O unit e.g. 170-AMM-090-00
4 CPU adapter e.g. 171-CCC-960-20-IEC
5 Hub or Switch

Defining Drops

To define drops proceed as follows in the **PLC Configuration** window:

Step	Action
1	Select **PLC Selection**. **Response:** The **PLC selection** dialog is opened.
2	Select the PLC family **Momentum** and CPU **171-CCC-960-20-IEC**. Use **OK** return to the **PLC Configuration** window.
3	Select **I/O Map**. **Response:** The **I/O Map** dialog is opened and the first drop is automatically entered in the table.
4	Select the drop from the **Drop** column. Select the **Edit...** command button. **Response:** You reach the module map.

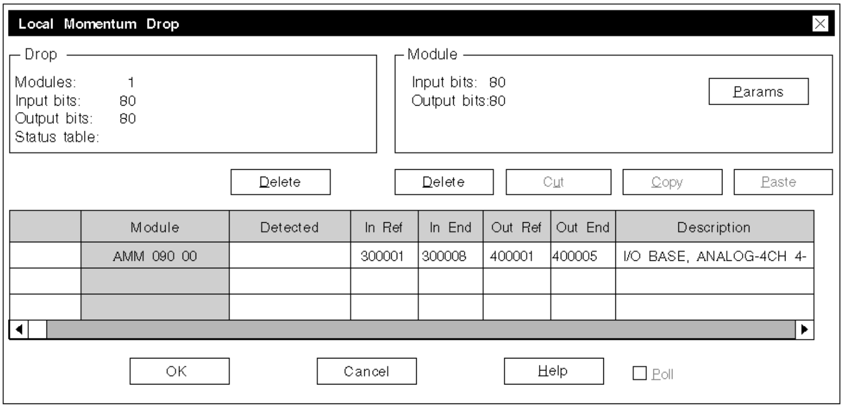
Mapping Modules and Specifying I/O References

To map the modules and specify the address ranges proceed as follows in the **Local Quantum Drop** dialog:

Step	Action
1	Select the **Module →...** column. **Response:** The **I/O Module Selection** dialog is opened.
2	From the **Category** column, select the **<all>** option. **Response:** All modules are listed in the **Modules** column.
3	Select from the column **Modules**, the module**AMM-090-00**. Exit the dialog with **OK**. **Response:** The module is inserted in the I/O map.
4	In the **In Ref** and **Out Ref** columns, set the start references for the input and output modules. **Note:** Discrete Input References have the prefix 1 (e.g. 100001), Coil References have the prefix 0 (e.g. 000001), Input Register References have the prefix 3 (e.g. 300001) and Output Register References have the prefix 4 (e.g. 400001). **Response:** The end reference (column **In.End.** or **Out.End**) of the available address range is automatically entered.

Dialog display

Following module mapping and I/O reference specification the dialog looks like this:



Local Momentum Drop

Drop

Modules: 1
Input bits: 80
Output bits: 80
Status table:

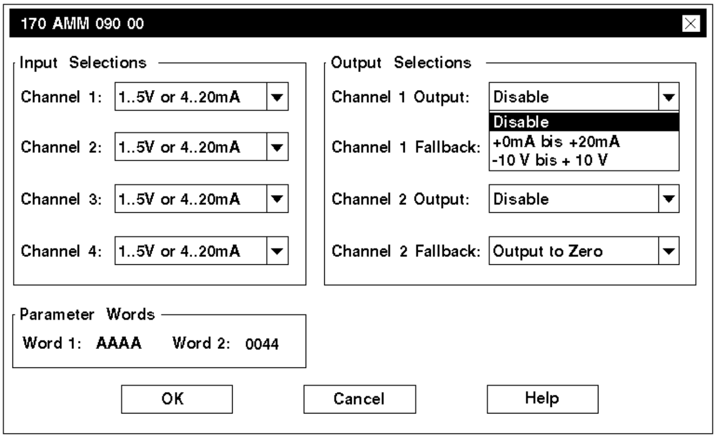
Module

Input bits: 80
Output bits:80

	Module	Detected	In Ref	In End	Out Ref	Out End	Description
	AMM 090 00		300001	300008	400001	400005	I/O BASE, ANALOG-4CH 4-

Set module parameters

To set parameters for the individual modules, proceed as follows in the **Local Momentum Drop** dialog:

Step	Action
1	Select the **Params** command button. **Response:** The **170-AMM-090-00** dialog is opened.
2	Select the signal states for the input and output channels from the list boxes and exit the dialog using **OK**. **Note:** Help with this can be obtained from the corresponding help text in the parameter dialog. Dialog display  170 AMM 090 00 Input Selections Channel 1: 1..5V or 4..20mA Channel 2: 1..5V or 4..20mA Channel 3: 1..5V or 4..20mA Channel 4: 1..5V or 4..20mA Output Selections Channel 1 Output: Disable Disable +0mA bis +20mA -10 V bis + 10 V Channel 1 Fallback: Channel 2 Output: Disable Channel 2 Fallback: Output to Zero Parameter Words Word 1: AAAA Word 2: 0044

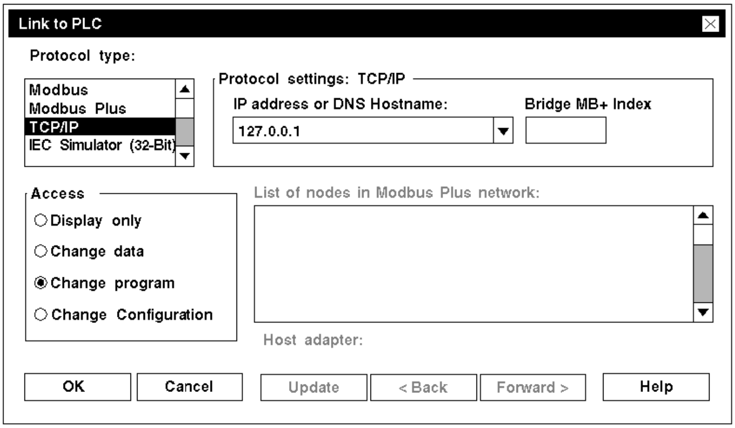
Create online connection

Introduction

This chapter describes how a link is created between the programming device and the Ethernet bus system.

Creating a link

For the link between the programming device and the Ethernet bus system use the Concept main menu **Online** and proceed as follows.

Step	Action
1	Select menu command**Link...**. **Response:** The **Link to PLC** dialog box opens.
2	From the list **Protocol type** select the link **TCP/IP**. **Response:** The zone **Protocol settings** alters for the TCP/IP settings.
3	In the text box IP address or DNS hostname enter the IP address of the Ethernet network card (PCI card). **Note:** Make sure that the address in Concept matches the address in Network settings of the operating system *(see page 976)*. **Response:** An online link exists between the programming device and the Ethernet bus system, and all bus nodes are displayed in the list. Dialog display  Link to PLC Protocol type: TCP/IP Protocol settings: TCP/IP IP address or DNS Hostname: 127.0.0.1 Bridge MB+ Index: Access: ○ Display only ○ Change data ● Change program ○ Change Configuration Modbus Modbus Plus TCP/IP IEC Simulator (32-Bit) List of nodes in Modbus Plus network: Host adapter: OK Cancel Update < Back Forward > Help
4	Exit the dialog using **OK**.

Convert Projects/DFBs/Macros

F

Converting projects/DFBs

At a Glance

The four main steps for converting projects/DFBs are as follows:

Step	Action
1	Exporting projects/DFBs/macros within the earlier version of Concept, see *Exporting project/DFB/macro (earlier version of Concept), page 992*.
2	For information on installing the new version of Concept, see *Installing new versions of Concepts, page 993*.
3	For information on importing projects/DFBs/macros, see *Importing project/DFB/macro, page 993*.
4	For information on editing projects/DFBs/macros, see *Editing the project/DFB/macro, page 993*.

Converting EFBs

⚠ **CAUTION**

Risk of losing data

If user-defined EFBs are being used in the project (EFBs which have been created manually), the current version of the EFB toolkit must be used to convert them (**File →Concept library…**). The Concept converter is not able to convert user-defined EFBs.

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

Exporting project/DFB/macro (earlier version of Concept)

The procedure for exporting projects/DFBs/macros is as follows:

⚠ **CAUTION**

Risk of losing data

The following steps must be performed in the **EARLIER** version of Concept. The new version of Concept may only be installed once all existing projects have been exported.

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

Step	Action
1	Start the Concept converter.
2	From **File** →**Export...** open the menu to select the export range.
3	Select the required export range: ● **Project with used DFBs:** All project information including the DFBs and data structures used within the project (derived data types) will be exported. ● **Project with all DFBs + macros:** All project information including all the DFBs and data structures (derived data types) will be exported. ● **Project without DFBs:** All project information including all data structures (derived data types), but excluding DFBs and macros will be exported. ● **Single DFB with used DFBs/single macro:** Only the selected DFB/macro will be exported. **Reaction:** The select export data dialog box will be opened.
4	Different file extensions must be selected depending on the element to be exported: ● **Exporting projects:** From the Format list select the extension .prj. ● **Exporting DFBs:** From the Format list select the extension .dfb. ● **Exporting macros:** From the Format list select the extension .mac.
5	Select the project / DFB / macro and confirm with **OK**. **Reaction:** The project/DFBs/macros/data structures (derived data types) will be contained in the current directory as an ASCII data file (.asc).
6	Quit the Concept converter with **File** →**Quit**.

Installing new versions of Concepts

⚠ CAUTION
Risk of losing data
Only install the NEW version of Concept if you have performed the previous steps.
Failure to follow these instructions can result in injury or equipment damage.

Follow the procedure described in the "Installation" chapter of the installation instructions.

Importing project/DFB/macro

The procedure for importing projects/DFBs/macros is as follows:

Step	Action
1	Start the Concept converter.
2	From **File** →**Import...** open the select import projects/DFBs/macros dialog box.
3	Select the project/DFB/macro (data file format .asc) and confirm with **OK**. **Reaction:** The project/DFBs/macros/data structures will be contained in the current directory as Concept data files.
4	Quit the Concept converter with **File** →**Quit**.

Editing the project/DFB/macro

Start the Concept/Concept DFB and edit the project/DFBs/macros/data structures in the usual way.

Concept ModConnect

G

Introduction

This chapter describes how to integrate third party modules into the Concept I/O map and how to remove it.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
G.1	Introduction	996
G.2	Integration of Third Party Modules	997
G.3	Use of third party module in Concept	1000

G.1 Introduction

Introduction

Overview

Information on hardware and I/O modules is stored in the Concept System Information Database (SysInfDb). This database is maintained and updated by Schneider and included with every Concept release.Nevertheless, Concept is able to support new I/O modules without having to wait for a new release. That's where the ModConnect Tool comes in - it takes a textual module description (MDC) and adds this information into the SysInfDb. This means that supplier of a new I/O module, who wants this module to be available in Concept, must also deliver an MDC file which describes the characteristics of this module.

Once installed, the I/O modules have the same functionality as existing Schneider Automation modules. This includes the ability to set module parameters and to display an online help.

For the installation of new modules, the third party module manufacturer has to supply a disk which contains a specific MDC file and the help information.

NOTE: The MDC file is dependent on the version of Concept so if you upgrade your Concept version, make sure you get also an upgraded version of your previously used MDC files. You will have to reinstall them.

G.2 Integration of Third Party Modules

Introduction

This chapter describes the procedures which have to be used in Concept ModConnect in order to integrate third party modules into Concept or to remove it.

What's in this Section?

This section contains the following topics:

Topic	Page
Integrating new Modules	998
Removing Modules	999

Integrating new Modules

Precondition

The specific MDC file for the new module has to be available.

Integrating new Modules

For integrating new modules, proceed the following steps:

Step	Action
1	For starting the application select **ModConnect Tool** in the Concept programm group. **Reaction:** Concept ModConnect displays its main window. If any Modules have been installed, a lis of installed modules is shown.
2	Copy the MDC file and the help file supplied with module to the Concept installation path.
3	Select **File** →**Open Installation File..**. **Reaction:** A dialog for selection the specific MDC file is opened.
4	Set the correct path to the MDC file and select it (e.g. SAMPLE.MDC). Confirm with **OK**. **Reaction:** The path including the name of the MDC file is now displayed in the **Select Module** dialog along with the defined modules.
5	Select the module you want to add and click **Add Module** or in the case of multiple entries click on the **Add All** button. You may additionally click the **Browse** button to return to the **Open file** dialog where you can select another .MDC for evaluation.
6	Click on the **Close** button to return to the main window. **Reaction:** The main window will now be displayed with the module information appearing in the **Imported Modules in Concept Database** window. By clicking on the added module (to select it) the module details are shown. With **Help** → **Help on Module** the help of the selected module can be displayed.
7	Select **File** →**Save Changes** to save the changes data base.
8	Select **File** →**Exit** for terminating Concept ModConnect. **Reaction:** The installed modules are now avaiable in the Concept I/O map *(see page 1000)*.

Upgrate of Concept

NOTE: The MDC-File is dependent on the version of Concept so if you upgrade your Concept version, make sure you get also an upgraded version of your previously used MDC files. You will have to reinstall them.

Removing Modules

Removing Modules

For removing modules, proceed the following steps:

Step	Action
1	For starting the application select **ModConnect Tool** in the Concept programm group. **Reaction:** Concept ModConnect displays its main window with a lis of the installed modules.
2	Select the module you want to remove and select **File** →**Remove selected Module**. **Reaction:** The **Confirm IOModule Removal** dialog is displayed.
3	Selecting **OK**, causes the removal of the module from Concept. **Reaction:** The module is no longer listed in the main window of Concept ModConnect or in the **I/O Module Selection** list box of Concept. **Note:** When removing modules. If the module has been used in existing Concept projects, the integrity of these projects will be compromised.
4	Select **File** →**Save Changes** to save the changes data base.
5	Select **File** →**Exit** for terminating Concept ModConnect. **Reaction:** The installed modules are now avaiable in the Concept I/O map *(see page 1000)*.

G.3 Use of third party module in Concept

Use of Third Party Modules in Concept

Precondition

The modules have to be installed according to the procedure *Integrating new Modules, page 998*.

Insert module to I/O Map

To insert a module to the I/O map, proceed the following steps:

Step	Action
1	Start Concept.
2	Open the configurator with **Project →Configurator**.
3	Open the I/O map with **I/O map... →Edit...**.
4	Open the **I/O Module Selection** dialog by clicking on **...** at the **Module** column. **Reaction:** The third party modules appear in the **Other** column.
5	Select the module by clicking. **Reaction:** A short description appear at the top of the dialog. You may press the **Help on Module** button to display the module's help file supplied by the vendor.
6	Click on **OK** (or doubleclick on the module) to insert the module the the I/O map. **Reaction:** The **I/O Module Selection** dialog is cloes and the selected module is inserted in the I/O map.
7	For entering the module's parameters (if available), select the **Rack-Slot** column of the module and click on the **Params** button. **Reaction:** The parameter screen for the selected dialog is opened.
8	Set the parameters for the module and confirm with **OK**.
9	Enter the input and output references for the module.
10	Confirm the I/O map with **OK** and save the project with **File →Save project**.

Convertion of Modsoft Programs

H

Introduction

This information provides you with the necessary process required to change previously generated Modsoft derived Ladder Logic programs into the Concept environment.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Introduction	1002
How to Convert a Modsoft Program	1004
Exceptions	1005

Introduction

Overview

For the convertion of an existing Modsoft program to a valid Concept 984 Ladder Logic project the Mosdsoft Converter is used. The Modsoft Converter provides current Modsoft users with a migration path to the 984 Ladder Logic for Windows environment. The Modsoft Converter requires no previous knowledge of the Concept programming environment. The term project is synonymous with a Modsoft program.

Starting the Modsoft Converter

Windows 98, Windows 2000 or Windows NT allows you to run the program from the **Start** menu, by selecting **Modsoft Converter** in the Concept programm group.

The Modsoft .ENV File

For the convertion the Modsoft .ENV file is needed. The .ENV file contains all the file information pertaining to the Modsoft program.

The Modsoft .ENV file contains the following files:
- **.CFG** Configuration file
- **.PRG** Ladder Logic file
- **.PCM** Network comments
- **.PCT** Network comments
- **.ASC** ASCII file
- **.USL** User Loadables
- **.RFD** Reference presets set by the user in the Modsoft Reference Data Editor
- **.REF** Reference contents contained in the PLC, from an upload
- **.RSF** Reference symbols

The convert process requires the .CFG file to be present in the .ENV file. If it does not exist, an dialog is displayed indicating that the .ENV file does not reference a .CFG file. All other files are optional.

By forcing you to enter the Modsoft *.ENV filename, some of the validation is avoided that would otherwise be required if you were allowed to enter a *.PRG and *.CFG name separately, i.e. Loadables (DX, User and EXE), state ram and builtin functions.

The first 5 lines of the .ENV file contain the path to the .ENV file. Be sure that the location of the .ENV file is as it is described; otherwise the conversion seems to be OK, but does not create a valid Concept application.

Incompatibilities

Due to differences in "address calculations in the configuration table" between Modsoft 2.6 and Concept 2.2 or later, the same Modsoft program loaded in a PLC and converted using the Modsoft Converter will cause a configuration miscompare in certain page zero locations. This will not affect the validity of the converted program.

Invalid PLC Types

If the Modsoft PLC type is not legal for Concept, a **PLC Selection** dialog box occurs and you have to select a compatible PLC. Please note that in this case the I/O map and other configuration elements will be defaulted or cleared.

Handling of SY/MAX

SY/MAX programs converted to Modsoft file format will migrate to the Quantum PLC type. The Modsoft Convert utility can then bring the SY/MAX program into Concept.

Modsoft Version

The Convert utility handles Modsoft file format supported in revision 2.2 or greater.

Handling of SFC and Macros

Modsoft does allow the user to save a Ladder Logic program that consists of undefined elements, and Concept needs to resolve those elements. The Modsoft Ladder Logic program is converted without performing any validity checks against the Configuration. When the Modsoft *.prg file contains either SFC or Macros the convert process is aborted and an dialog is displayed informing you to return to Modsoft and use **Segment Status →Commands →Convert to File**. This process expands the Macros and translates the SFC elements.

Handling of I/O Map

Modsoft sets a default I/O map size of 512. Concept does not, but calculates the size as required. Uploading a Controller that has been downloaded with Modsoft will cause a miscompare. You are allowed to continue.

Handling of References

Modsoft can have two types of reference data or none at all. There exists online reference data information (RAM) if you have uploaded from the PLC. There are also references defined using the offline Reference Data Editor. When both types of data exist in the .env file, the convert utility first imports the online references then overlays the offline reference data.

How to Convert a Modsoft Program

Precondition

For converting a Modsoft program the Modsoft .ENV file *(see page 1002)* is necessary. The .ENV file contains all the file information pertaining to the Modsoft program. Once selected the conversion takes place and you are prompted to a **Save as** dialog.

How to Convert a Mosdsoft Programm

For converting a Modsoft programm, proceed the following steps:

Step	Action
1	Open the Modsoft Converter.
2	Select **File** → **Convert...**.
3	Select the drive and the directory, where to find the Modsoft .ENV file. (The file will be found in the Modsoft program directory, e.g. C:\Modsoft\Programs.)
4	Pick the file from the list.
5	Start the convertion with **Convert**. **Reaction:** The convertion is started. ● A convert progress dialog is displayed after the validity checks on the *.ENV file are performed. The first line of the dialog indicates the section currently being converted and the second line indicates progress as it pertains to the whole convert process. ● f any errors, such as **Out of memory**, **Out of disk space** or **File access errors**, occur during the convert process, an error dialog is displayed. ● An operation completed error free results in the automatic display of the **Save as Concept project** dialog. The default name of the project, displayed in the Save project dialog, is the *.ENV filename prompt.
6	You can then change the project name and the directory in which Concept project will be saved. **Reaction:** If the project name selected already exists a confirmation dialog is displayed. **Note:** Saving the Modsoft converted program as a Concept project does not have to be done at this time, you can still save using the **File** →**Save project as** menu item.

Exceptions

Description

0x and 1x references in a Modsoft program are converted to a Located Variable with data type BOOL in Concept. This data type is compatible with the use of these references.

However, 3x and 4x are converted to integer.

NOTE: This straight conversion precludes both Modsoft bit defination and floating point types.

Example

If you have the following defined in **Modsoft**:

REF	BIT	SYMBOL	DESCRIPTOR
000001		located_0x_boolean	located 0x boolean descriptor
100001		located_1x_boolean	located 1x boolean descriptor
300001	/16	bit_16_of_3000001	16th bit of 300001 descriptor
400100		incoming_integer	incoming integer descriptor
400200		outgoing_interger	outgoing flt32 descriptor
400300	/ 1	bit_1_of_400300	bit 1 of 400300 descriptor

A conversion of the above to **Concept** using the Convert program yields:

Variable Name	Data Type	Address	Comment
located_0x_boolean	BOOL	000001	located 0x boolean descriptor
located_1x_boolean	BOOL	100001	located 1x boolean descriptor
bit_16_of_3000001	INT	300001	16th bit of 300001 descriptor
incoming_integer	INT	400100	incoming integer descriptor
outgoing_interger	INT	400200	outgoing flt32 descriptor
bit_1_of_400300	INT	400300	bit 1 of 400300 descriptor

Modsoft and 984 References

I

Introduction

This chapter contains the Modsoft and 984 References.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Modsoft Keys with Concept Equivalents	1008
Modsoft Function Compatibility	1010

Modsoft Keys with Concept Equivalents

Keys

NOTE: When possible, the **Ctrl** key is used in place of the Modsoft **Alt** key.

Table of keys:

Funtion	Modsoft 2.x Key	Concept Key
Normally open contact	' or "	same
Coil	**(** or **[**	same
Normally closed contacts	**/** or \	same
Horizontal short	**=**	same
Vertical short	**I**	same
Negative transitional contact	**Alt**+**N**	**N**
Positive transitional contact	**Alt**+**P**	**P**
Inserting a function block by name	**Alt**+**F**	**Ctrl**+**F**
Copy element(s)	**Alt**+**F3**	**Ctrl**+**C**
Delete element(s)	**Alt**+**F4** or **Del**	**Ctrl**+**X** or **Del**
Paste	**Alt**+**F5**	**Ctrl**+**V**
Offset references	**Alt**+**F6**	**Ctrl**+**H**
Search	**Alt**+**F7**	**F3**
Search next	**Alt**+**F8**	**F6** When online in direct mode, Concept uses a nonmodal dialog with accelerators for search previous and search next.
Network comments	**Alt**+**C**	**Ctrl**+**M**
Goto network	**Alt**+**G**	**Ctrl**+**G**
Insert network	**Alt**+**I**	**Ctrl**+**I**
Append network	**Alt**+**A**	**Ctrl**+**A**
Trace	**Alt**+**T**	**Ctrl**+**T**
Retrace	**Alt**+**B**	**Ctrl**+**B** or **Ctrl**+**T**
Dx zoom	**Alt**+**Z**	**Ctrl**+**D**
Goto node (1,1) of active network	**Home**	same
Goto node (7,11) of active network	**End**	same
Goto first network in current segment	**Ctrl**+**Home**	same
Goto last network in current segment	**Ctrl**+**End**	same

Funtion	Modsoft 2.x Key	Concept Key
Insert equation	**Ins**	**Ctrl**+**Q**
Append	-	**Ctrl**+**A**
Append equation	-	**Ctrl**+**U**
Delete current network	-	**Ctrl**+**K**
Copy to the clipboard	-	**Ctrl**+**C**
Undo	-	**Ctrl**+**Z**
Closing an mdi child window	-	**Ctrl**+**F4**
Switching to the next open mdi child window	-	**Ctrl**+**F6**

Status Line Values

These Concept keys change the status line display value of the currently selected reference:

A	ASCII
H	Hexidecimal
D	Decimal (signed)
U	Decimal (unsigned)
R	Real
L	Long (32 bit)
S	Short (16 bit)

Modsoft Function Compatibility

Not Supported Features

The following Modsoft functions are **not** supported in Concept:

- Macros/macro programming
- SFC (use IEC SFC instead)
- Search of comments

User Interface Difference

Concept is an MS-Windows based application. Modsoft is a DOS based application. Concept uses MS-Windows user interface standards and practices. Functions of Concept with 984 Ladder editor are based on the pre-existing functions of Concept.

There are no exact similarities of specific user actions required to perform Concept tasks as compared to Modsoft tasks.

Constant Sweep

Concept has no off line selection to set the constant sweep mode. This mode is available from the **Online Control Panel**.

Once constant sweep has been set in the controller, you can upload the controller and save the project. The constant sweep settings will be retained in the project. If this project is downloaded, the constant sweep settings will be set.

NOTE: Any changes to the controller configuration cause the constant sweep settings to be reset, i.e, constant sweep is disabled whenever the controller configuration changes. Follow the steps above to reenable constant sweep.

How to Start the Constant Sweep

To set constant sweep before starting the controller, follow these steps:

Step	Action
1	Create your configuration and program logic, offline.
2	Download your program to the controller. When the dialog appears asking `Do you want to start the controller?`"click on the **No** button.
3	From the **Online** menu, choose **Online Control Panel**.
4	Set the constant sweep mode and sweep time.
5	Start the controller.

Presettings when using Modbus Plus for startup

J

Overview

This chapter provides a brief description of the presettings when using Modbus Plus for first startup.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Installing the SA85/PCI85 with Windows 98/2000/XP	1012
Installing the SA85/PC185 in Windows NT	1016
Installing the Modbus Plus Driver in Windows 98/2000/NT	1018
Virtual MBX Driver for 16 bit application capability with Windows 98/2000/NT	1019
MBX Driver for connection between ModConnect Host interface adapters and 32 bit applications with Windows 98/2000/NT	1020
Remote MBX - Driver for Remote Operation	1021
Ethernet MBX - Driver for Modbus Plus Function via TCP/IP	1022
Establishing the hardware connection.	1024

Installing the SA85/PCI85 with Windows 98/2000/XP

Introduction

A Modbus Plus connection can be made using the SA85 or PCI85 adapters.

The difference between the adapters is in the bus used:

- SA85 for ISA Bus
- PCI85 for PCI Bus

While the Modbus Node Address and Memory Based Address for the SA85 is set directly on the card with the DIP switches, the address for the PCI85 is made during the configuration in Windows.

SA85 Hardware settings

Carry out the following steps to configure the Hardware settings for the SA85:

Step	Action
1	Enter the Modbus node location (Modbus Plus Port Location) and the memory based address in SA85 (see documentation "IBM Host Based Devices").
2	Install the SA85 as described in the "IBM Host Based Devices" documentation.

PCI85 Installation

Install the PCI85 (416 NHM 300 30 or 416 NHM 300 32) as described in the "Modbus Plus PCI-85 Interface Adapter" 890 USE 162 00 documentation.

Driver installation

Install the Virtual MBX driver and then the MBX or Remote MBX driver.

See also:

- *Virtual MBX Driver for 16 bit application capability with Windows 98/2000/NT, page 1019*
- *MBX Driver for connection between ModConnect Host interface adapters and 32 bit applications with Windows 98/2000/NT, page 1020*
- *Remote MBX - Driver for Remote Operation, page 1021*

Configuration

Carry out the following steps to configure the adapter after installing the driver:

Step	Action
1	Open the Control Panel (**Start** →**Settings** →**Control Panel**).
2	**Windows XP:** Select the **Printer and other Hardware** icon.
3	**Windows XP:** Select the **System** icon.
4	Select the **Hardware** icon. **Result:** The hardware wizard is called.
5	Select the **Next** command button.
6	**Windows 98:** Select the option **Yes (Recommended)**. **Windows 2000/XP:** Select the option **Add/Troubleshoot a device**. Select the **Next** command button. **Result:** Hardware detection is started.
7	**Only for Windows 98:** Select the **Next** command button. **Result:** The hardware detection status is displayed.
8	**Only for Windows 98:** Select the **Next** command button. **Result:** All hardware types are displayed in a list.
9	Select the hardware type **MBX Devices for Modicon Networks**, and press the **Next** command button. **Result:** The database with driver information is created.
10	Select the **SA85-000** adapter or **PCI85-000** and press the **Next** command button. **Result:** A memory range is automatically defined.
11	Select the **Next** command button. **Result:** The automatically assigned device number and request mode (20 ms) is displayed.
12	Select the **Next** command button. **Result:** The software for the new hardware components is installed.
13	Select the **Next** command button. **Result:** You are asked to shutdown the computer.
14	Press the **No** command button. **Result:** The adapter is configured with the default settings.

Win 98: Edit configuration

Carry out the following steps to edit the configuration using Windows 98 after the first configuration:

Step	Action
1	Open the Control Panel (**Start** →**Settings** →**Control Panel**).
2	Select the **System** icon. **Result:** The **System Properties** window is opened.
3	Select the **Device Manager** tab.
4	Select the **SA85-000** adapter or **PCI85-000** and press the **Properties** command button. **Result:** The **SA85-000/PCI85-000 Adapter Properties** window is opened.
5	Select the **Device Settings** tab.
6	Make the changes as required. (See also the Help file LMBX9X on the driver CD.)
7	Select the **Resources** tab to change the memory area.
8	Use the **OK** command button to exit the window. **Result:** The changes are accepted by the system.

Win 2000/XP: Edit configuration

Carry out the following steps to edit the configuration using Windows 2000/XP after the first configuration:

Step	Action
1	Open the Control Panel (**Start** →**Settings** →**Control Panel**).
2	**Windows XP:** Select the **Printer and other Hardware** icon.
3	Select the **System** icon. **Result:** The **System Properties** window is opened.
4	Select the **Hardware** tab.
5	Select the **Device Manager...** command button. **Result:** The **Device Manager** window is opened.
6	Select the **Network adapter** →**SA85-000** or **PCI85-000**.
7	Select the **Properties** command button. **Result:** The **SA85-000PCI85-000 Adapter Properties** window is opened.
8	Select the **Device Settings** tab.
9	Make the changes as required. (See also the Help file LMBX9X on the driver CD.)
10	Select the **Resources** tab to change the memory area.
11	Use the **OK** command button to exit the window. **Result:** The changes are accepted by the system.

Peer Cop functions

Several parameter settings must be made to enable Peer Cop communication via the adapter. The Peer Cop function is disabled by default, and should only be enabled if your applications require Peer Cop communication.

To enable and set parameters for Peer Cop communication, start with the first steps as with "Edit Configuration". In the **SA85-000/PCI85-000 Adapter Properties** window, select the **Peer Cop** tab and make your settings as desired.

Installing the SA85/PC185 in Windows NT

Introduction

A Modbus Plus connection can be made using the SA85 or PCI85 adapters.

The difference between the adapters is in the bus used:
- SA85 for ISA Bus
- PCI85 for PCI Bus

While the Modbus Node Address and Memory Based Address for the SA85 is set directly on the card with the DIP switches, the address for the PCI85 is made during the configuration in Windows.

SA85 Hardware Settings

Carry out the following steps to set the SA85 hardware settings:

Step	Action
1	Set the Modbus node address (Modbus Plus Port Address) and the memory based address on the SA85 (see documentation "IBM Host Based Devices").
2	Install the SA85 as described in the "IBM Host Based Devices" documentation.

PCI85 Installation

Install the PCI85 (416 NHM 300 30 or 416 NHM 300 32) as described in the "Modbus Plus PCI-85 Interface Adapter" 890 USE 162 00 documentation.

Installing drivers

Install the Virtual MBX driver and then the MBX or Remote MBX driver.

Also see:
- *Virtual MBX Driver for 16 bit application capability with Windows 98/2000/NT, page 1019*
- *MBX Driver for connection between ModConnect Host interface adapters and 32 bit applications with Windows 98/2000/NT, page 1020*
- *Remote MBX - Driver for Remote Operation, page 1021*

Configuration

Carry out the following steps to configure the adapter after installing the driver:

Step	Action
1	In the start menu, open the folder **WinConX/MBXDriver** (**Start** →**Program** → **WinConX**).
2	Double-click on the **MBX Driver Configuration** icon. **Result:** The dialog box **MBX Driver configuration** is opened.
3	In the **Device Configuration** register, click on the command button **New**. **Result:** A list box will appear in the **Device type** column.
4	Select the option **SA85** or **PCI85** from the list. **Result:** The dialog box **SA85 configuration** is opened.
5	Make the following settings. (also see Help file LMBX9X on the driver CD.) **Note:** With the PCI85 you enter the Modbus Node address in the **Node** list box.
6	Exit the dialog box by clicking **Close**. **Result:** The settings are accepted by the system.

Edit configuration

Carry out the following steps to edit the configuration after the first configuration:

Step	Action
1	In the start menu, open the folder **WinConX/MBXDriver** (**Start** →**Program** → **WinConX**).
2	Double-click on the **MBX Driver configuration** icon. **Result:** The dialog box **MBX Driver configuration** is opened.
3	Select SA85 from the **Device configuration** register.
4	Click on the command button **Edit**. **Result:** The **SA85 configuration** dialog box is opened.
5	Make the following changes. (also see Help file LMBX9X on the driver CD.)
6	Exit the dialog box by clicking **Close**. **Result:** The settings are accepted by the system.

Peer Cop functionality

Several parameter settings must be made to enable Peer Cop communication via the adapter. The Peer Cop function is deactivated as standard, and should only be enabled if your application requires Peer Cop communication.

To enable and set parameters for Peer Cop communication, start with the first steps as with "Edit SA85 Configuration". In the dialog box **SA85 configuration**, select the **Peer Cop** register and make your settings.

Installing the Modbus Plus Driver in Windows 98/2000/NT

Introduction

In order to use the Modbus Plus communication, you must first install the CyberLogic MBX driver for Windows 98/2000/NT version >=4.20 (+ Service Release 1 for Windows 2000)

The following drivers are available on the CD "MBX Driver Suite v4.20":

Driver	Operating system
MBX Driver *MBX Driver for connection between ModConnect Host interface adapters and 32 bit applications with Windows 98/2000/NT, page 1020*	Windows 98/2000/NT
Virtual MBX Driver *Virtual MBX Driver for 16 bit application capability with Windows 98/2000/NT, page 1019*	Windows 98/2000/NT
Remote MBX Driver *Remote MBX - Driver for Remote Operation, page 1021*	Windows 98/2000/NT
Ethernet MBX Driver *Ethernet MBX - Driver for Modbus Plus Function via TCP/IP, page 1022*	Windows NT

Installation

Carry out the following steps to install the Modbus Plus driver:

Step	Action
1	Start Windows.
2	Insert the CD "MBX Driver Suite ver. 4.20".
3	Select the **Start** →**Execute** command.
4	Enter the CD drive and **:\SETUP** in the command line.
5	Confirm with **OK**.
6	Follow the onscreen instructions. **Response:** After installation the WinConX program with all installed drivers is created in the Start Menu.

Configuration

Configuration occurs automatically after installing the driver. To make changes to the configuration, open the dialog to be edited from the **WinConX** →**xxx MBX Driver** Start Menu, by double clicking on the **xxx MBX Configuration Editor** symbol.

Virtual MBX Driver for 16 bit application capability with Windows 98/2000/NT

Introduction

Installing the Virtual MBX driver guarantees the run capability of all 16 bit DOS or Windows 3.x NETLIB/NetBIOS compatible applications in their original binary form in Windows 98/2000/NT.

NOTE: A detailed description of this driver can be found in the VMBX9X or VMBXNT Help file on the "MBX Driver Suite ver4.20" CD.

Preconditions

In order for the Virtual MBX driver to function correctly, additional drivers must be installed.

The following additional drivers can be installed to enhance the Virtual MBX driver's run capability:

Driver	Operating system	Application
MBX	Windows 98/2000/NT	Driver for Modbus Plus Host interface adapter
Remote MBX	Windows 98/2000/NT	Driver for accessing remote nodes on the Modbus Plus and Ethernet network
Ethernet MBX	Windows NT	Driver for Modbus Plus Emulation via TCP/IP

Installation

The virtual MBX driver software for Windows 98/2000 and Windows NT is included along with other drivers, on the CD "MBX Driver Suite ver4.20".

Installation is done by Autorun when the CD is inserted or can be started manually (CD drive:\SETUP.EXE). Select the driver to be installed from the main menu. You will then be taken through the installation step by step. The driver is then configured.

Configuration

NOTE: To guarantee a connection to Concept (= 16-Bit-Application), in the **Virtual MBX Driver Configuration →16-bit Windows Applications** dialog, check the **Support 16-bit Windows Applications** checkbox.

MBX Driver for connection between ModConnect Host interface adapters and 32 bit applications with Windows 98/2000/NT

Introduction

The installation of the MBX driver guarantees the connection between the MODConnect Host interface adapter and 32 bit applications with Windows 98/2000/NT. This driver also supports the program interfaces MBXAPI and NETLIB. This means that practically all Modbus Plus compatible software programs can be operated via Modbus, Modbus Plus and Ethernet networks, without having to make changes. This also includes 32 bit Windows 98/2000/NT applications and 16 bit old DOS/Windows applications.

NOTE: A detailed description of the driver is included on the CD "MBX Driver Suite ver4.20" in the Help file LMBX9X or LMBXNT.

Hardware support

The MBX driver operates either in Interrupt or Polled mode.

It supports the following ModConnect Host interface adapter:
- ISA
- EISA
- MCA
- PC card (PCMCIA)

Remote connection

The MBX driver includes the MBX Remote Server. This enables remote nodes to access local MBX devices (including the Host interface adapter) via any Windows 98/2000/NT compatible network. Also see *Remote MBX - Driver for Remote Operation, page 1021*.

Installation

The MBX driver software for Windows 98/2000/NT is included along with other drivers, on the CD "MBX Driver Suite ver4.20".

The installation is carried out by Autorun when the CD is inserted or can be manually started(CD drive:\SETUP.EXE). Select the driver to be installed from the main menu. You will then be taken through the installation step by step. The driver is then configured.

Remote MBX - Driver for Remote Operation

Introduction

The installation of the remote MBX driver allows remote connection of applications operated on remote station client nodes. Remote station access of the Modbus Plus network takes place using a standard LAN (Local Area Network).

This driver also unites applications that support the program interfaces MBXAPI and NETLIB.

NOTE: A detailed description of the driver is included on the CD "MBX Driver Suite ver4.20" in the Help file RMBX9X or RMBXNT.

Preconditions

This connection is only made if your programming device is a node on the MBX Remote Server. Also install one of these drivers because the MBX and Ethernet MBX drivers include the MBX Remote Server.

Installation

The remote MBX driver software for Windows 98/2000/NT is included along with other drivers, on the CD "MBX Driver Suite ver4.20".

The installation is carried out by Autorun when the CD is inserted or can be manually started(CD drive:\SETUP.EXE). Select the driver to be installed from the main menu. You will then be taken through the installation step by step. The driver is then configured.

Configuration

The configuration of the remote MBX driver is presently the same as the configuration of the other MBX drivers. The remote MBX driver is operated as a remote client node, which does not require a physical host interface adapter. Therefore the driver configuration also includes the creation of logical devices (MBX Remote Client), which refer to the physical devices found on the server node.

Ethernet MBX - Driver for Modbus Plus Function via TCP/IP

Introduction

The installation can only be carried out in Windows NT.

When the Ethernet MBX driver is installed, Modbus Plus function is emulated via TCP/IP. This driver also supports the program interfaces MBXAPI and NETLIB. This means that practically all Modbus Plus compatible software programs immediately have access to TCP/IP based communication without having to make changes. This also includes 32 bit Windows 98/2000/NT applications and 16 bit old DOS/Windows applications.

NOTE: A detailed description of the driver is included on the CD "MBX Driver Suite ver4.20" in the Help file EMBXNT.

Winsock API

When using Winsock API, the Ethernet MBX driver can solve certain critical problems created by the Winsock interface.

For example: TCP Port 502 can only receive one process with incoming messages. If several applications attempt to receive unexpected messages, a conflict occurs. The Ethernet MBX driver eliminated this problem by acting as global dispatcher for these messages. When using the slave path, Concept in Modbus Plus determines that several (up to 256) applications refer to these unexpected messages and execute them simultaneously.

Advantage of using the driver

The most important advantages when using the driver via the Winsock API are:

- Changes are no longer needed for existing NETLIP/NetBIOS/MBXAPI compatible applications. End user and developer software investments are completely secured.
- Consistent management and dispatching of unexpected messages, which prevents overlaps between various products on the same system.
- Complete functionality of TCP/IP communication, while protecting existing NETLIP/NetBIOS/MBXAPI standards.
  For example: Ethernet TCP/IP communication requires a identifier address in the form of an IP address, and a message contains an identifier index byte. The Ethernet MBX driver protects this functionality.
- Working with TCP/IP communication is an advantage for software developers not experienced with the complicated Winsock API.
- A single program model for software developers handles communication in Modbus, Modbus Plus and Ethernet TCP/IP networks.
- Increased compatibility with various products. Winsock API is more oriented towards developer executed, TCP/IP strategies in a slightly different manner and can create compatibility problems in various products.
- Compatible with all MBX products. How e.g. the Virtual MBX driver for use of old 16-bit DOS/Windows applications and the MBX driver which dispatches messages between Ethernet, Modbus, Modbus Plus and remote MBX nodes.

Remote connection

The Ethernet MBX driver includes the MBX Remote Server. This enables remote nodes to access local MBX devices (including Ethernet MBX devices) via any Windows compatible network. The remote client can be a Windows 98/2000/NT node with the remote MBX driver installed. Also see *Remote MBX - Driver for Remote Operation, page 1021*.

Installation

The Ethernet MBX driver software for Windows NT is included along with other drivers, on the CD "MBX Driver Suite ver4.20".

The installation is carried out by Autorun when the CD is inserted or can be manually started(CD drive:\SETUP.EXE). Select the driver to be installed from the main menu. You will then be taken through the installation step by step. The driver is then configured.

Establishing the hardware connection.

Introduction

NOTE: Please refer to the "Modbus Protocol Reference Guide" for a detailed description of the hardware setup.

Procedure

To establish the hardware connection, do the following:

Step	Action
1	Set a unique Modbus node address for the CPU using the rotary switch on the back of the module.
2	Note the Modbus node address set on the CPU's sliding cover.
3	Connect the CPU to the SA85 interface with a Modbus Plus cable. **Result:** The flash interval on the CPU "Modbus +" display changes from 3 flashes per second with a pause to 6 without a pause.

Presettings when using Modbus for startup

K

Overview

The chapter provides a brief description of the presettings when using Modbus for startup.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Interface Settings in Windows 98/2000/XP	1026
Interface Settings in Windows NT	1028
Setting up the hardware connection	1029
Transfer problems	1030

Interface Settings in Windows 98/2000/XP

Win 98: Interface settings

Carry out the following steps to configure the interface in Windows 95/98/2000:

Step	Action
1	Select the **My Computer** icon. **Result:** All available objects are displayed.
2	Select the **Control Panel** icon. **Result:** All available objects are displayed.
3	Select the **System** icon. **Result:** The **System Properties** dialog box is opened.
4	Select the **Device Manager** tab.
5	Select **Ports (COM and LPT)**. **Result:** The branches**Communications Port (COMx)** and **Printer Port (LPTx)** are displayed.
6	Select **Communications Port (COMx)**. **Result:** The **Communications Port (COMx) Properties** dialog box is opened.
7	Select the **Port Settings** tab.
8	Select the **Advanced...** command button. **Result:**The **Advanced Settings** dialog box is opened.
9	Check the **Use FIFO buffers** check box. **Note:** Using the FIFO(First In First Out) buffer requires a serial port with 16550 compatible UART (Universal Asynchronous Receiver Transmitter).
10	Use the slider to modify the receive and send buffer by setting both buffers to the maximum size.
11	Close all dialog boxes using the **OK** command button.

Win 2000/XP: Interface settings

Carry out the following steps to configure the interface in Windows 2000/XP:

Step	Action
1	Select the **My Computer** icon. **Result:** All available objects are displayed.
2	Select the **Control Panel** icon. **Result:** All available objects are displayed.
3	**Only with Win XP:** Select the **Printer and other Hardware** icon.
4	Select the **System** icon. **Result:**The **System Properties** dialog box is opened.
5	Select the **Hardware** tab.
6	Select the **Device Manager...** command button. **Result:** The **Device Manager** window is opened.
7	Select **Ports (COM and LPT)**. **Result:** The branches**ECP Printer Port (LPT1)** and **Communications Port (COMx)** are displayed.
8	Select **Communications Port (COMx)**. **Result:** The **Communications Port (COMx) Properties** dialog box is opened.
9	Select the **Port Settings** tab.
10	Select the **Advanced...** command button. **Result:** The **Advanced settings for COMx** dialog box is opened.
11	Check the **Use FIFO buffers** check box. **Note:** Using the FIFO(First In First Out) buffer requires a serial port with 16550 compatible UART (Universal Asynchronous Receiver Transmitter).
12	Use the slider to modify the receive and send buffer by setting both buffers to the maximum size.
13	Close all dialog boxes using the **OK** command button.

Interface Settings in Windows NT

Interface setting

Carry out the following steps to set the interface in Windows NT:

Step	Action
1	Double-click on the **My Computer** icon. **Response:** All available objects are displayed.
2	Double-click on the **Control Panel** icon. **Response:** All available objects are displayed.
3	Double-click on the **Connections** icon. **Response:** The **Connections** dialog box is opened.
4	Select the connection to be set in the list box and click on the command button **Settings...**. **Response:** The **COMx Settings** dialog box is opened.
5	Click on the command button **Extended...**. **Response:** The **Advanced Settings for COMx** dialog box is opened.
6	Activate the check box **FIFO activated**. **Note:** Using the FIFO(First In First Out) buffer requires a serial port with 16550 compatible UART (Universal Asynchronous Receiver Transmitter).
7	Close all dialogs with **OK**.

Setting up the hardware connection

Introduction

NOTE: Please refer to the "Modbus Protocol Reference Guide" for a detailed description of the hardware setup.

Procedure

To establish the hardware connection, do the following:

Step	Action
1	Set a unique Modbus node address for the CPU using the rotary switch on the back of the module.
2	Note the Modbus node address set on the CPU’s sliding cover.
3	Connect the Modbus interface CPU to the PC serial COM interface with a Modbus cable.

Transfer problems

Introduction

Communication errors can occur when loading the EXEC file. Communication, made via the COM interface with Windows, depends on several factors. These factors include the programming device clock speed, the communication software and the other programs (or applications) that are used in the system.

Check list for transfer problems

Refer to the following check list if transfer problems occur:

Step	Action
1	Check that no other applications are running in the background. Another application running in the background can mean that the active communication application in the foreground cannot receive information fast enough.
2	Check that the programming device is running at the highest possible clock speed. Some programming devices can prolong the lifetime of the buffer battery with lower speeds. Look in the documentation for you computer.
3	Use a serial connector with a 16550A Universal Asynchronous Receiver Transmitter (UART). Windows uses the buffering capability of these connections so that Windows data transfer applications can reach higher speeds even on slower computers.

RTU transfer problems

If sporadic errors occur during data transfer, transfer cannot be carried out successfully with RTU mode. If this is the case, select ASCII mode. (See Quantum/Compact/Momentum/Atrium first startup.)

Startup when using Modbus with the EXECLoader

L

Overview

This chapter describes loading executive data (EXEC) onto the PLC with the EXECLoader program.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Quantum first startup with EXECLoader	1032
Compact first startup with EXECLoader	1036
Momentum first startup for IEC with EXECLoader	1040
Momentum first startup for LL984 with EXECLoader	1044

Quantum first startup with EXECLoader

Introduction

This section describes the first startup of Quantum when used with Modbus.

The first startup is subdivided into 5 main sections:

- *Start EXECLoader, page 1032*
- *Define Modbus interface, page 1032*
- *Protocol settings, page 1033*
- *Select EXEC file, page 1034*
- *Load EXEC file, page 1034*

Start EXECLoader

The procedure for launching EXECLoaders is as follows:

Step	Action
1	Open the Concept start menu. **Response:** All installed Concept programs are displayed as symbols.
2	Click on the symbol with the identifier **EXECLoader**. **Response:** The EXECLoader program is started.
3	Click on the command **Next**, as soon as you have read the information on the page. **Response:** The dialog **Communication Protocol** is opened.

Define Modbus interface

Carry out the following steps to set the Modbus interface:

Step	Action
1	Select the Option button **Modbus RTU (RS232)** for the RTU transfer mode. Select the Option button **Modbus ASCII (RS232)** for the ASCII transfer mode. **Note:** Data transfer can only take place if you have configured the same transfer mode (RTU or ASCII) on the CPU (using a button on the front of the module).
2	Click on the command button **Next**. **Response:** The dialog **Modbus Target →RTU/ASCII mode** is opened.
3	Use the command button **COM Port Settings...** to open the dialog **COM Properties**.
4	Use the list field **Connect using:** to select the programming cable interface on the PC (default setting is **COM1**).
5	Use the list field **Bits per second:** to select the Baudrate (default is **9600**).
6	Use the list field **Parity:** to select the parity (default is **EVEN**).

Step	Action
7	Use the list field **Stop Bits** to select the Stop bits (default is **1**).
8	Click on the command button **OK**. **Response:** The dialog is closed and you return to the dialog **Modbus Target** → **RTU/ASCII Mode**.

Protocol settings

Carry out the following steps to set the Modbus protocol:

Step	Action
1	Click on the command button **Scan**. **Response:** The nodes on the Modbus network are read and displayed graphically in the left window. A green point in the graphic indicates that the CPU is in RUN mode. To stop the CPU continue as described in step 3.
2	Double-click on the read network node in the graphical display. **Response:**The Modbus address of the node is automatically entered in the textfeld **Modbus Address**.
3	Click the right mouse button in the left window. **Response:** A context menu with individual PLC commands is opened.
4	If the "Run" display is lit on the CPU, stop the program using the command **Stop PLC**. **Response:** A message window appears where you can click **OK** to confirm stopping the CPU.
5	Click the command button **OK**, to confirm stopping the CPU. **Response:** You return to the dialog **Modbus Target** →**RTU mode**, and the green point disappears from the graph.
6	Activate the check box **Bridge**, if the connection to the Modbus node should be made via a Modbus Plus network using a Modbus bridge.
7	Press the appropriate Option button for your system (**PLC**, **Direct Device**, **Local Head**, **Remote I/O Drop**).
8	Click on the command button **Next**. **Response:** The **Operation** dialog is opened.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Press the Option button **Transfer EXEC to Device**.
2	Click on the command button **Browse...**. **Response:** The Concept directory is opened in a standard window.
3	Double-click on the DAT directory. **Response:** All available*.BIN files are displayed.
4	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Quantum PLC Types, page 1101*. **Response:** The selected *.BIN file is displayed in **File name:** text field.
5	Click on the command button **Open**. **Response:** You return to the dialog **Operation**, and the path to the selected *.BIN file is displayed in the **Filename** text field.
6	Click on the command button **Next**. **Response:** The dialog **File and Device Info** is opened. Information is provided here about the selected *.BIN file and also about the PLC.
7	Click on the command button **Next**. **Response:** The **Summary** dialog is opened. This gives you an overview of the settings made for you to check.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Click on the command button **Transfer**. **Response:** A message box appears warning you that all data available on the PLC will be lost, and the configuration and program must be reloaded on the PLC.
2	Click on the command button **Yes**, to continue the transfer. **Response:** The **Progress** dialog is opened. This gives information about the progress of the transfer in a progress bar and text.
3	Click **Close** once the transfer is complete. **Response:** The dialog is closed, and you return to the dialog **Summary**.
4	Click on the command button **Close**, to close the EXECLoader.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	slow flashing
Modbus	lit (with some interruptions)
Modbus +	3x flashes with interruptions

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	not lit
Modbus	not lit
Modbus +	3x flashes with interruptions

NOTE: The three flash sequence Modbus + display idicates that no communication is present on the bus. This is displayed on Modbus by a non-lit Modbus display. The Modbus display is lit again once connection is made with Concept.

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

Compact first startup with EXECLoader

Introduction

This section describes the first startup of Compact when used with Modbus.

The first startup is subdivided into 5 main sections:

- *Start EXECLoader, page 1036*
- *Define Modbus interface, page 1036*
- *Protocol settings, page 1037*
- *Select EXEC file, page 1038*
- *Load EXEC file, page 1038*

Start EXECLoader

The procedure for launching EXECLoaders is as follows:

Step	Action
1	Open the Concept start menu. **Response:** All installed Concept programs are displayed as symbols.
2	Click on the symbol with the identifier **EXECLoader**. **Response:** The EXECLoader program is started.
3	Click on the command **Next**, as soon as you have read the information on the page. **Response:** The dialog **Communication Protocol** is opened.

Define Modbus interface

Carry out the following steps to set the Modbus interface:

Step	Action
1	Select the Option button **Modbus RTU (RS232)** for the RTU transfer mode. Select the Option button **Modbus ASCII (RS232)** for the ASCII transfer mode. **Note:** Data transfer can only take place if you have configured the same transfer mode (RTU or ASCII) on the CPU (using a button on the front of the module).
2	Click on the command button **Next**. **Response:** The dialog **Modbus Target →RTU/ASCII mode** is opened.
3	Use the command button **COM Port Settings...** to open the dialog **COM Properties**.
4	Use the list field **Connect using:** to select the programming cable interface on the PC (default setting is **COM1**).
5	Use the list field **Bits per second:** to select the Baudrate (default is **9600**).
6	Use the list field **Parity:** to select the parity (default is **EVEN**).

Step	Action
7	Use the list field **Stop Bits** to select the Stop bits (default is **1**).
8	Click on the command button **OK**. **Response:** The dialog is closed and you return to the dialog **Modbus Target** → **RTU/ASCII Mode**.

Protocol settings

Carry out the following steps to set the Modbus protocol:

Step	Action
1	Click on the command button **Scan**. **Response:** The nodes on the Modbus network are read and displayed graphically in the left window. A green point in the graphic indicates that the CPU is in RUN mode. To stop the CPU continue as described in step 3.
2	Double-click on the read network node in the graphical display. **Response:**The Modbus address of the node is automatically entered in the textfeld **Modbus Address**.
3	Click the right mouse button in the left window. **Response:** A context menu with individual PLC commands is opened.
4	If the "Run" display is lit on the CPU, stop the program using the command **Stop PLC**. **Response:** A message window appears where you can click **OK** to confirm stopping the CPU.
5	Click the command button **OK**, to confirm stopping the CPU. **Response:** You return to the dialog **Modbus Target** →**RTU mode**, and the green point disappears from the graph.
6	Activate the check box **Bridge**, if the connection to the Modbus node should be made via a Modbus Plus network using a Modbus bridge.
7	Press the appropriate Option button for your system (**PLC**, **Direct Device**, **Local Head**, **Remote I/O Drop**).
8	Click on the command button **Next**. **Response:** The **Operation** dialog is opened.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Press the Option button **Transfer EXEC to Device**.
2	Click on the command button **Browse...**. **Response:** The Concept directory is opened in a standard window.
3	Double-click on the DAT directory. **Response:** All available*.BIN files are displayed.
4	Click on the *.BIN file CTSX201D. **Response:** The selected *.BIN file is displayed in **File name:** text field.
5	Click on the command button **Open**. **Response:** You return to the dialog **Operation**, and the path to the selected *.BIN file is displayed in the **Filename** text field.
6	Click on the command button **Next**. **Response:** The dialog **File and Device Info** is opened. Information is provided here about the selected *.BIN file and also about the PLC.
7	Click on the command button **Next**. **Response:** The **Summary** dialog is opened. This gives you an overview of the settings made for you to check.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Click on the command button **Transfer**. **Response:** A message box appears warning you that all data available on the PLC will be lost, and the configuration and program must be reloaded on the PLC.
2	Click on the command button **Yes**, to continue the transfer. **Response:** The **Progress** dialog is opened. This gives information about the progress of the transfer in a progress bar and text.
3	Click **Close** once the transfer is complete. **Response:** The dialog is closed, and you return to the dialog **Summary**.
4	Click on the command button **Close**, to close the EXECLoader.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	slow flashing
Modbus	lit (with some interruptions)
Modbus +	3x flashes with interruptions

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	not lit
Modbus	not lit
Modbus +	3x flashes with interruptions

NOTE: The three flash sequence Modbus + display idicates that no communication is present on the bus. This is displayed on Modbus by a non-lit Modbus display. The Modbus display is lit again once connection is made with Concept.

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

Momentum first startup for IEC with EXECLoader

Introduction

This section describes the first startup of Momentum for IEC when used with Modbus.

The first startup is subdivided into 5 main sections:

- *Start EXECLoader, page 1040*
- *Define Modbus interface, page 1040*
- *Protocol settings, page 1041*
- *Select EXEC file, page 1042*
- *Load EXEC file, page 1042*

Start EXECLoader

The procedure for launching EXECLoaders is as follows:

Step	Action
1	Open the Concept start menu. **Response:** All installed Concept programs are displayed as symbols.
2	Click on the symbol with the identifier **EXECLoader**. **Response:** The EXECLoader program is started.
3	Click on the command **Next**, as soon as you have read the information on the page. **Response:** The dialog **Communication Protocol** is opened.

Define Modbus interface

Carry out the following steps to set the Modbus interface:

Step	Action
1	Select the Option button **Modbus RTU (RS232)** for the RTU transfer mode. Select the Option button **Modbus ASCII (RS232)** for the ASCII transfer mode. **Note:** Data transfer can only take place if you have configured the same transfer mode (RTU or ASCII) on the CPU (using a button on the front of the module).
2	Click on the command button **Next**. **Response:** The dialog **Modbus Target →RTU/ASCII mode** is opened.
3	Use the command button **COM Port Settings...** to open the dialog **COM Properties**.
4	Use the list field **Connect using:** to select the programming cable interface on the PC (default setting is **COM1**).
5	Use the list field **Bits per second:** to select the Baudrate (default is **9600**).
6	Use the list field **Parity:** to select the parity (default is **EVEN**).

Step	Action
7	Use the list field **Stop Bits** to select the Stop bits (default is **1**).
8	Click on the command button **OK**. **Response:** The dialog is closed and you return to the dialog **Modbus Target** → **RTU/ASCII Mode**.

Protocol settings

Carry out the following steps to set the Modbus protocol:

Step	Action
1	Click on the command button **Scan**. **Response:** The nodes on the Modbus network are read and displayed graphically in the left window. A green point in the graphic indicates that the CPU is in RUN mode. To stop the CPU continue as described in step 3.
2	Double-click on the read network node in the graphical display. **Response:**The Modbus address of the node is automatically entered in the textfeld **Modbus Address**.
3	Click the right mouse button in the left window. **Response:** A context menu with individual PLC commands is opened.
4	If the "Run" display is lit on the CPU, stop the program using the command **Stop PLC**. **Response:** A message window appears where you can click **OK** to confirm stopping the CPU.
5	Click the command button **OK**, to confirm stopping the CPU. **Response:** You return to the dialog **Modbus Target** →**RTU mode**, and the green point disappears from the graph.
6	Activate the check box **Bridge**, if the connection to the Modbus node should be made via a Modbus Plus network using a Modbus bridge.
7	Press the appropriate Option button for your system (**PLC**, **Direct Device**, **Local Head**, **Remote I/O Drop**).
8	Click on the command button **Next**. **Response:** The **Operation** dialog is opened.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Press the Option button **Transfer EXEC to Device**.
2	Click on the command button **Browse...**. **Response:** The Concept directory is opened in a standard window.
3	Double-click on the DAT directory. **Response:** All available*.BIN files are displayed.
4	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Momentum PLC Types, page 1103*. **Response:** The selected *.BIN file is displayed in **File name:** text field.
5	Click on the command button **Open**. **Response:** You return to the dialog **Operation**, and the path to the selected *.BIN file is displayed in the **Filename** text field.
6	Click on the command button **Next**. **Response:** The dialog **File and Device Info** is opened. Information is provided here about the selected *.BIN file and also about the PLC.
7	Click on the command button **Next**. **Response:** The **Summary** dialog is opened. This gives you an overview of the settings made for you to check.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Click on the command button **Transfer**. **Response:** A message box appears warning you that all data available on the PLC will be lost, and the configuration and program must be reloaded on the PLC.
2	Click on the command button **Yes**, to continue the transfer. **Response:** The **Progress** dialog is opened. This gives information about the progress of the transfer in a progress bar and text.
3	Click **Close** once the transfer is complete. **Response:** The dialog is closed, and you return to the dialog **Summary**.
4	Click on the command button **Close**, to close the EXECLoader.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Run	slow flashing
COM ACT	lit (with some interruptions)
Modbus +	3x flashes with interruptions

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Run	not lit
COM ACT	not lit
Modbus +	3x flashes with interruptions

NOTE: The three flash sequence Modbus + display idicates that no communication is present on the bus. This is displayed on Modbus by a non-lit COM ACT display. The COM ACT display is lit again once connection is made with Concept.

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

Momentum first startup for LL984 with EXECLoader

Introduction

This section describes the first startup of Momentum for LL984 when used with Modbus.

NOTE: Loading the EXEC file for LL984 is not necessary with a new computer, since it is preloaded in the the CPUs Flash RAM. Loading the EXEC file for LL984 is only necessary if you have already loaded the EXEC file for IEC, and now wish to change.

You should always check to see if a new EXEC version has been released in the meantime. This information and the current EXEC file can be found on our website at www.schneiderautomation.com. You can see the currently loaded version of the EXEC file in Concept using the **Online →Online controller...** menu command.

The first startup is subdivided into 5 main sections:

- *Start EXECLoader, page 1044*
- *Define Modbus interface, page 1045*
- *Protocol settings, page 1045*
- *Select EXEC file, page 1046*
- *Load EXEC file, page 1047*

Start EXECLoader

The procedure for launching EXECLoaders is as follows:

Step	Action
1	Open the Concept start menu. **Response:** All installed Concept programs are displayed as symbols.
2	Click on the symbol with the identifier **EXECLoader**. **Response:** The EXECLoader program is started.
3	Click on the command **Next**, as soon as you have read the information on the page. **Response:** The dialog **Communication Protocol** is opened.

Define Modbus interface

Carry out the following steps to set the Modbus interface:

Step	Action
1	Select the Option button **Modbus RTU (RS232)** for the RTU transfer mode. Select the Option button **Modbus ASCII (RS232)** for the ASCII transfer mode. **Note:** Data transfer can only take place if you have configured the same transfer mode (RTU or ASCII) on the CPU (using a button on the front of the module).
2	Click on the command button **Next**. **Response:** The dialog **Modbus Target →RTU/ASCII mode** is opened.
3	Use the command button **COM Port Settings...** to open the dialog **COM Properties**.
4	Use the list field **Connect using:** to select the programming cable interface on the PC (default setting is **COM1**).
5	Use the list field **Bits per second:** to select the Baudrate (default is **9600**).
6	Use the list field **Parity:** to select the parity (default is **EVEN**).
7	Use the list field **Stop Bits** to select the Stop bits (default is **1**).
8	Click on the command button **OK**. **Response:** The dialog is closed and you return to the dialog **Modbus Target → RTU/ASCII Mode**.

Protocol settings

Carry out the following steps to set the Modbus protocol:

Step	Action
1	Click on the command button **Scan**. **Response:** The nodes on the Modbus network are read and displayed graphically in the left window. A green point in the graphic indicates that the CPU is in RUN mode. To stop the CPU continue as described in step 3.
2	Double-click on the read network node in the graphical display. **Response:**The Modbus address of the node is automatically entered in the textfeld **Modbus Address**.
3	Click the right mouse button in the left window. **Response:** A context menu with individual PLC commands is opened.
4	If the "Run" display is lit on the CPU, stop the program using the command **Stop PLC**. **Response:** A message window appears where you can click **OK** to confirm stopping the CPU.
5	Click the command button **OK**, to confirm stopping the CPU. **Response:** You return to the dialog **Modbus Target →RTU mode**, and the green point disappears from the graph.

Step	Action
6	Activate the check box **Bridge**, if the connection to the Modbus node should be made via a Modbus Plus network using a Modbus bridge.
7	Press the appropriate Option button for your system (**PLC**, **Direct Device**, **Local Head**, **Remote I/O Drop**).
8	Click on the command button **Next**. **Response:** The **Operation** dialog is opened.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Press the Option button **Transfer EXEC to Device**.
2	Click on the command button **Browse...**. **Response:** The Concept directory is opened in a standard window.
3	Double-click on the DAT directory. **Response:** All available*.BIN files are displayed.
4	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Momentum PLC Types, page 1103*. **Response:** The selected *.BIN file is displayed in **File name:** text field.
5	Click on the command button **Open**. **Response:** You return to the dialog **Operation**, and the path to the selected *.BIN file is displayed in the **Filename** text field.
6	Click on the command button **Next**. **Response:** The dialog **File and Device Info** is opened. Information is provided here about the selected *.BIN file and also about the PLC.
7	Click on the command button **Next**. **Response:** The **Summary** dialog is opened. This gives you an overview of the settings made for you to check.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Click on the command button **Transfer**. **Response:** A message box appears warning you that all data available on the PLC will be lost, and the configuration and program must be reloaded on the PLC.
2	Click on the command button **Yes**, to continue the transfer. **Response:** The **Progress** dialog is opened. This gives information about the progress of the transfer in a progress bar and text.
3	Click **Close** once the transfer is complete. **Response:** The dialog is closed, and you return to the dialog **Summary**.
4	Click on the command button **Close**, to close the EXECLoader.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Run	slow flashing
COM ACT	lit (with some interruptions)
Modbus +	3x flashes with interruptions

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Run	not lit
COM ACT	not lit
Modbus +	3x flashes with interruptions

NOTE: The three flash sequence Modbus + display idicates that no communication is present on the bus. This is displayed on Modbus by a non-lit COM ACT display. The COM ACT display is lit again once connection is made with Concept.

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

Startup when using Modbus with DOS Loader

M

Overview

This chapter describes loading executive data (EXEC) onto the PLC with the DOS Loader program.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Quantum first startup with DOS Loader	1050
Compact first startup with DOS Loader	1053
Momentum first startup for IEC with DOS Loader	1056
Momentum first startup for LL984 with DOS Loader	1059

Quantum first startup with DOS Loader

Introduction

This section describes the first startup of Quantum when used with Modbus.

The first startup is subdivided into 5 main sections:
- *Start DOS Loader, page 1050*
- *Define Modbus interface, page 1050*
- *Protocol settings, page 1051*
- *Select EXEC file, page 1051*
- *Load EXEC file, page 1051*

Start DOS Loader

The procedure for launching DOS Loaders is as follows:

Step	Action
1	If the CPU display "Run" is lit, stop the program with Concept (in main menu **Online**). **Response:** The "Run" display is no longer lit; the "Ready" is now lit.
2	Open the directory DAT (CONCEPT\DAT).
3	Double-click on the file LOADER.EXE. **Response:** The installation program for the Executive file (EXEC) is started.
4	Select the option **Communication Parameters**. **Response:** The dialog box **Communication setup** is opened.
5	Select the **Modbus**option. **Response:** The **Modbus communication setup** window is opened.

Define Modbus interface

Carry out the following steps to set the Modbus interface:

Step	Action
1	Select the programming cable interface on the PC (default setting is **COM1**).
2	Select the Baudrate (default is **9600**).
3	Select the parity (default is **EVEN**).
4	Select the Option button **RTU - 8 Bits** for the RTU transfer mode. Select the option **ASCII -7 Bits** for the ASCII transfer mode.
5	Select the Stop bits (default is **1**). **Note:** Data transfer can only take place if you have configured the same transfer mode (ASCII or RTU) on the CPU (using a button on the front of the module).

Protocol settings

Carry out the following steps to set the Modbus protocol:

Step	Action
1	Enter in **Enter PLC Address:** the node address set on the CPU (using a rotary switch on the back of the module).
2	Select the **TARGET PATH 0**option.
3	Select the **ACCEPT CHANGES**option. **Response:** You return to the main menu.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Select the option **File Selection**. **Response:** The **File Selection** window is opened.
2	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Quantum PLC Types, page 1101*.
3	Confirm your selection with RETURN. **Response:** You return to the main menu.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Select the option **Load File To PLC**. **Response:** The **Loading Process** window is opened, and the warning "The PROGRAM contents of the device being loaded could be lost after loading a new Executive. The CONTENTS stored in the Micro H H P will always be lost after loading a new Executive. Would you like to continue (Y/N) ? N" is displayed.
2	Acknowledge the warning with **Y**. **Response:** The message "Node failed to enter normal mode" appears.
3	The DOWNLOAD PROGRESS window appears which shows the transfer rate. **Response:** After the transfer is complete, the message "Download Operation Successful" appears.
4	Confirm the message with RETURN. **Response:** You return to the main menu.
5	Select the **Exit Program**option. **Response:** The DOS Loader is exited.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	slow flashing
Modbus	lit (with some interruptions)
Modbus +	3x flashes with interruptions

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	not lit
Modbus	not lit
Modbus +	3x flashes with interruptions

NOTE: The three flash sequence Modbus + display idicates that no communication is present on the bus. This is displayed on Modbus by a non-lit Modbus display. The Modbus display is lit again once connection is made with Concept.

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

Compact first startup with DOS Loader

Introduction

This section describes the first startup of Compact when used with Modbus.

The first startup is subdivided into 5 main sections:
- *Start DOS Loader, page 1053*
- *Define Modbus interface, page 1053*
- *Protocol settings, page 1054*
- *Select EXEC file, page 1054*
- *Load EXEC file, page 1054*

Start DOS Loader

The procedure for launching DOS Loaders is as follows:

Step	Action
1	If the CPU display "Run" is lit, stop the program with Concept (in main menu **Online**). **Response:** The "Run" display is no longer lit; the "Ready" is now lit.
2	Open the directory DAT (CONCEPT\DAT).
3	Double-click on the file LOADER.EXE. **Response:** The installation program for the Executive file (EXEC) is started.
4	Select the option **Communication Parameters**. **Response:** The dialog box **Communication setup** is opened.
5	Select the **Modbus**option. **Response:** The **Modbus communication setup** window is opened.

Define Modbus interface

Carry out the following steps to set the Modbus interface:

Step	Action
1	Select the programming cable interface on the PC (default setting is **COM1**).
2	Select the Baudrate (default is **9600**).
3	Select the parity (default is **EVEN**).
4	Select the Option button **RTU - 8 Bits** for the RTU transfer mode. Select the option **ASCII -7 Bits** for the ASCII transfer mode.
5	Select the Stop bits (default is **1**). **Note:** Data transfer can only take place if you have configured the same transfer mode (ASCII or RTU) on the CPU (using a button on the front of the module).

Protocol settings

Carry out the following steps to set the Modbus protocol:

Step	Action
1	Enter in **Enter PLC Address:** the node address set on the CPU (using a rotary switch on the back of the module).
2	Select the **TARGET PATH 0**option.
3	Select the **ACCEPT CHANGES**option. **Response:** You return to the main menu.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Select the option **File Selection**. **Response:** The **File Selection** window is opened.
2	Select the *.BIN file CTSX201D.
3	Confirm your selection with RETURN. **Response:** You return to the main menu.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Select the option **Load File To PLC**. **Response:** The **Loading Process** window is opened, and the warning "The PROGRAM contents of the device being loaded could be lost after loading a new Executive. The CONTENTS stored in the Micro H H P will always be lost after loading a new Executive. Would you like to continue (Y/N) ? N" is displayed.
2	Acknowledge the warning with **Y**. **Response:** The message "Node failed to enter normal mode" appears.
3	The DOWNLOAD PROGRESS window appears which shows the transfer rate. **Response:** After the transfer is complete, the message "Download Operation Successful" appears.
4	Confirm the message with RETURN. **Response:** You return to the main menu.
5	Select the **Exit Program**option. **Response:** The DOS Loader is exited.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	slow flashing
Modbus	not lit
Modbus +	fast flashing

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	not lit
Modbus	not lit
Modbus +	fast flashing

NOTE: The three flash sequence Modbus + display idicates that no communication is present on the bus. This is displayed on Modbus by a non-lit Modbus display. The Modbus display is lit again once connection is made with Concept.

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

Momentum first startup for IEC with DOS Loader

Introduction

This section describes the first startup of Momentum for IEC when used with Modbus.

The first startup is subdivided into 5 main sections:

- *Start DOS Loader, page 1056*
- *Define Modbus interface, page 1056*
- *Protocol settings, page 1057*
- *Select EXEC file, page 1057*
- *Load EXEC file, page 1057*

Start DOS Loader

The procedure for launching DOS Loaders is as follows:

Step	Action
1	If the CPU display "Run" is lit, stop the program with Concept (in main menu **Online**). **Response:** The "Run" display is no longer lit; the "Ready" is now lit.
2	Open the directory DAT (CONCEPT\DAT).
3	Double-click on the file LOADER.EXE. **Response:** The installation program for the Executive file (EXEC) is started.
4	Select the option **Communication Parameters**. **Response:** The dialog box **Communication setup** is opened.
5	Select the **Modbus**option. **Response:** The **Modbus communication setup** window is opened.

Define Modbus interface

Carry out the following steps to set the Modbus interface:

Step	Action
1	Select the programming cable interface on the PC (default setting is **COM1**).
2	Select the Baudrate (default is **9600**).
3	Select the parity (default is **EVEN**).
4	Select the Option button **RTU - 8 Bits** for the RTU transfer mode. Select the option **ASCII -7 Bits** for the ASCII transfer mode.
5	Select the Stop bits (default is **1**). **Note:** Data transfer can only take place if you have configured the same transfer mode (ASCII or RTU) on the CPU (using a button on the front of the module).

Protocol settings

Carry out the following steps to set the Modbus protocol:

Step	Action
1	Enter in **Enter PLC Address:** the node address set on the CPU (using a rotary switch on the back of the module).
2	Select the **TARGET PATH 0**option.
3	Select the **ACCEPT CHANGES**option. **Response:** You return to the main menu.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Select the option **File Selection**. **Response:** The **File Selection** window is opened.
2	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Momentum PLC Types, page 1103*.
3	Confirm your selection with RETURN. **Response:** You return to the main menu.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Select the option **Load File To PLC**. **Response:** The **Loading Process** window is opened, and the warning "The PROGRAM contents of the device being loaded could be lost after loading a new Executive. The CONTENTS stored in the Micro H H P will always be lost after loading a new Executive. Would you like to continue (Y/N) ? N" is displayed.
2	Acknowledge the warning with **Y**. **Response:** The message "Node failed to enter normal mode" appears.
3	The DOWNLOAD PROGRESS window appears which shows the transfer rate. **Response:** After the transfer is complete, the message "Download Operation Successful" appears.
4	Confirm the message with RETURN. **Response:** You return to the main menu.
5	Select the **Exit Program**option. **Response:** The DOS Loader is exited.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Run	slow flashing
COM ACT	lit (with some interruptions)
Modbus +	3x flashes with interruptions

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Run	not lit
COM ACT	not lit
Modbus +	3x flashes with interruptions

NOTE: The three flash sequence Modbus + display idicates that no communication is present on the bus. This is displayed on Modbus by a non-lit COM ACT display. The COM ACT display is lit again once connection is made with Concept.

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

Momentum first startup for LL984 with DOS Loader

Introduction

This section describes the first startup of Momentum for LL984 when used with Modbus.

NOTE: Loading the EXEC file for LL984 is not necessary with a new computer, since it is preloaded in the the CPUs Flash RAM. Loading the EXEC file for LL984 is only necessary if you have already loaded the EXEC file for IEC, and now wish to change.

You should always check to see if a new EXEC version has been released in the meantime. This information and the current EXEC file can be found on our website at www.schneiderautomation.com. You can see the currently loaded version of the EXEC file in Concept using the **Online →Online controller...** menu command.

The first startup is subdivided into 5 main sections:

- *Start DOS Loader, page 1059*
- *Define Modbus interface, page 1060*
- *Protocol settings, page 1060*
- *Select EXEC file, page 1060*
- *Load EXEC file, page 1061*

Start DOS Loader

The procedure for launching DOS Loaders is as follows:

Step	Action
1	If the CPU display "Run" is lit, stop the program with Concept (in main menu **Online**). **Response:** The "Run" display is no longer lit; the "Ready" is now lit.
2	Open the directory DAT (CONCEPT\DAT).
3	Double-click on the file LOADER.EXE. **Response:** The installation program for the Executive file (EXEC) is started.
4	Select the option **Communication Parameters**. **Response:** The dialog box **Communication setup** is opened.
5	Select the **Modbus**option. **Response:** The **Modbus communication setup** window is opened.

Define Modbus interface

Carry out the following steps to set the Modbus interface:

Step	Action
1	Select the programming cable interface on the PC (default setting is **COM1**).
2	Select the Baudrate (default is **9600**).
3	Select the parity (default is **EVEN**).
4	Select the Option button **RTU - 8 Bits** for the RTU transfer mode. Select the option **ASCII -7 Bits** for the ASCII transfer mode.
5	Select the Stop bits (default is **1**). **Note:** Data transfer can only take place if you have configured the same transfer mode (ASCII or RTU) on the CPU (using a button on the front of the module).

Protocol settings

Carry out the following steps to set the Modbus protocol:

Step	Action
1	Enter in **Enter PLC Address:** the node address set on the CPU (using a rotary switch on the back of the module).
2	Select the **TARGET PATH 0**option.
3	Select the **ACCEPT CHANGES**option. **Response:** You return to the main menu.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Select the option **File Selection**. **Response:** The **File Selection** window is opened.
2	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Momentum PLC Types, page 1103*.
3	Confirm your selection with RETURN. **Response:** You return to the main menu.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Select the option **Load File To PLC**. **Response:** The **Loading Process** window is opened, and the warning "The PROGRAM contents of the device being loaded could be lost after loading a new Executive. The CONTENTS stored in the Micro H H P will always be lost after loading a new Executive. Would you like to continue (Y/N) ? N" is displayed.
2	Acknowledge the warning with **Y**. **Response:** The message "Node failed to enter normal mode" appears.
3	The DOWNLOAD PROGRESS window appears which shows the transfer rate. **Response:** After the transfer is complete, the message "Download Operation Successful" appears.
4	Confirm the message with RETURN. **Response:** You return to the main menu.
5	Select the **Exit Program**option. **Response:** The DOS Loader is exited.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Run	slow flashing
COM ACT	lit (with some interruptions)
Modbus +	3x flashes with interruptions

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Run	not lit
COM ACT	not lit
Modbus +	3x flashes with interruptions

NOTE: The three flash sequence Modbus + display idicates that no communication is present on the bus. This is displayed on Modbus by a non-lit COM ACT display. The COM ACT display is lit again once connection is made with Concept.

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

Startup when using Modbus Plus with the EXECLoader

N

Overview

This chapter describes loading executive data (EXEC) onto the PLC with the EXECLoader.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Quantum first startup with EXECLoader	1064
Compact first startup with EXECLoader	1068
Atrium first startup with EXECLoader	1072
Momentum first startup for IEC with EXECLoader	1076
Momentum first startup for LL984 with EXECLoader	1080

Quantum first startup with EXECLoader

Introduction

This section describes the first startup of Quantum when used with Modbus Plus.

The first startup is subdivided into 5 main sections:
- *Start EXECLoader, page 1064*
- *Define SA85 adapter, page 1064*
- *Protocol settings, page 1065*
- *Select EXEC file, page 1066*
- *Load EXEC file, page 1066*

Start EXECLoader

The procedure for launching EXECLoaders is as follows:

Step	Action
1	Open the Concept start menu. **Response:** All installed Concept programs are displayed as symbols.
2	Click on the symbol with the identifier **EXECLoader**. **Response:** The EXECLoader program is started.
3	Click on the command **Next**, as soon as you have read the information on the page. **Response:** The dialog **Communication Protocol** is opened.

Define SA85 adapter

Carry out the following steps to define the LAN address set in the CONFIG.SYS file:

Step	Action
1	Select the option button **Modbus Plus**.
2	Click on the command button **Next**. **Response:** The **Modbus Plus Target** dialog is opened.
3	Select from the list **Devices Online:** the adapter address you set when installing the SA85 in the CONFIG.SYS file (Parameter /n).

Protocol settings

Carry out the following steps to set the Modbus Plus protocol settings:

Step	Action
1	Click on the command button **Scan**. **Response:** The nodes on the Modbus Plus network are read and displayed graphically in the left window. A green point in the graphic indicates that the CPU is in RUN mode. To stop the CPU continue as described in step 3.
2	Double-click on the read network node in the graphical display. **Response:** The Modbus address of the node is automatically entered in the textfeld In the text field **Modbus Plus Address**.
3	Click the right mouse button in the left window. **Response:** A context menu with individual PLC commands is opened.
4	If the "Run" display is lit on the CPU, stop the program using the command **Stop PLC**. **Response:** A message window appears where you can click **OK** to confirm stopping the CPU.
5	Click the command button **OK**, to confirm stopping the CPU. **Response:** You return to the **Modbus Plus Target** dialog, and the green point disappears from the graph.
6	Activate the check box **Bridge**, if the connection to the Modbus network node should be made via a Modbus network using a Modbus bridge.
7	Press the appropriate Option button for your system (**PLC**, **Local Head**, **Remote I/O Drop**).
8	Click on the command button **Next**. **Response:** The **Operation** dialog is opened.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Press the Option button **Transfer EXEC to Device**.
2	Click on the command button **Browse...**. **Response:** The Concept directory is opened in a standard window.
3	Double-click on the DAT directory. **Response:** All available*.BIN files are displayed.
4	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Quantum PLC Types, page 1101*. **Response:** The selected *.BIN file is displayed in **File name:** text field.
5	Click on the command button **Open**. **Response:** You return to the dialog **Operation**, and the path to the selected *.BIN file is displayed in the **Filename** text field.
6	Click on the command button **Next**. **Response:** The dialog **File and Device Info** is opened. Information is provided here about the selected *.BIN file and also about the PLC.
7	Click on the command button **Next**. **Response:** The **Summary** dialog is opened. This gives you an overview of the settings made for you to check.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Click on the command button **Transfer**. **Response:** A message box appears warning you that all data available on the PLC will be lost, and the configuration and program must be reloaded on the PLC.
2	Click on the command button **Yes**, to continue the transfer. **Response:** The **Progress** dialog is opened. This gives information about the progress of the transfer in a progress bar and text.
3	Click **Close** once the transfer is complete. **Response:** The dialog is closed, and you return to the dialog **Summary**.
4	Click on the command button **Close**, to close the EXECLoader.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	slow flashing
Modbus	not lit
Modbus +	fast flashing

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	not lit
Modbus	not lit
Modbus +	fast flashing

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

NOTE: If you recieve an error message, close Concept and start the BDRESET.EXE file (in the \Concept directory) to reset the SA85. Then start again from the first step.

Compact first startup with EXECLoader

Introduction

This section describes the first startup of Compact when used with Modbus Plus.

The first startup is subdivided into 5 main sections:
- *Start EXECLoader, page 1068*
- *Define SA85 adapter, page 1068*
- *Protocol settings, page 1069*
- *Select EXEC file, page 1070*
- *Load EXEC file, page 1070*

Start EXECLoader

The procedure for launching EXECLoaders is as follows:

Step	Action
1	Open the Concept start menu. **Response:** All installed Concept programs are displayed as symbols.
2	Click on the symbol with the identifier **EXECLoader**. **Response:** The EXECLoader program is started.
3	Click on the command **Next**, as soon as you have read the information on the page. **Response:** The dialog **Communication Protocol** is opened.

Define SA85 adapter

Carry out the following steps to define the LAN address set in the CONFIG.SYS file:

Step	Action
1	Select the option button **Modbus Plus**.
2	Click on the command button **Next**. **Response:** The **Modbus Plus Target** dialog is opened.
3	Select from the list **Devices Online:** the adapter address you set when installing the SA85 in the CONFIG.SYS file (Parameter /n).

Protocol settings

Carry out the following steps to set the Modbus Plus protocol settings:

Step	Action
1	Click on the command button **Scan**. **Response:** The nodes on the Modbus Plus network are read and displayed graphically in the left window. A green point in the graphic indicates that the CPU is in RUN mode. To stop the CPU continue as described in step 3.
2	Double-click on the read network node in the graphical display. **Response:** The Modbus address of the node is automatically entered in the textfeld In the text field **Modbus Plus Address**.
3	Click the right mouse button in the left window. **Response:** A context menu with individual PLC commands is opened.
4	If the "Run" display is lit on the CPU, stop the program using the command **Stop PLC**. **Response:** A message window appears where you can click **OK** to confirm stopping the CPU.
5	Click the command button **OK**, to confirm stopping the CPU. **Response:** You return to the **Modbus Plus Target** dialog, and the green point disappears from the graph.
6	Activate the check box **Bridge**, if the connection to the Modbus network node should be made via a Modbus network using a Modbus bridge.
7	Press the appropriate Option button for your system (**PLC**, **Local Head**, **Remote I/O Drop**).
8	Click on the command button **Next**. **Response:** The **Operation** dialog is opened.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Press the Option button **Transfer EXEC to Device**.
2	Click on the command button **Browse...**. **Response:** The Concept directory is opened in a standard window.
3	Double-click on the DAT directory. **Response:** All available*.BIN files are displayed.
4	Click on the *.BIN file CTSX201D. **Response:** The selected *.BIN file is displayed in **File name:** text field.
5	Click on the command button **Open**. **Response:** You return to the dialog **Operation**, and the path to the selected *.BIN file is displayed in the **Filename** text field.
6	Click on the command button **Next**. **Response:** The dialog **File and Device Info** is opened. Information is provided here about the selected *.BIN file and also about the PLC.
7	Click on the command button **Next**. **Response:** The **Summary** dialog is opened. This gives you an overview of the settings made for you to check.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Click on the command button **Transfer**. **Response:** A message box appears warning you that all data available on the PLC will be lost, and the configuration and program must be reloaded on the PLC.
2	Click on the command button **Yes**, to continue the transfer. **Response:** The **Progress** dialog is opened. This gives information about the progress of the transfer in a progress bar and text.
3	Click **Close** once the transfer is complete. **Response:** The dialog is closed, and you return to the dialog **Summary**.
4	Click on the command button **Close**, to close the EXECLoader.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	slow flashing
Modbus	not lit
Modbus +	fast flashing

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	not lit
Modbus	not lit
Modbus +	fast flashing

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

NOTE: If you recieve an error message, close Concept and start the BDRESET.EXE file (in the \Concept directory) to reset the SA85. Then start again from the first step.

Atrium first startup with EXECLoader

Introduction

This section describes the first startup of Atrium when used with Modbus Plus. The hardware requirements for loading EXEC files can be seen in the "Modicon TSX Atrium" manual.

The first startup is subdivided into 5 main sections:

- *Start EXECLoader, page 1072*
- *Define SA85 adapter, page 1072*
- *Protocol settings, page 1073*
- *Select EXEC file, page 1074*
- *Load EXEC file, page 1074*

Start EXECLoader

The procedure for launching EXECLoaders is as follows:

Step	Action
1	Open the Concept start menu. **Response:** All installed Concept programs are displayed as symbols.
2	Click on the symbol with the identifier **EXECLoader**. **Response:** The EXECLoader program is started.
3	Click on the command **Next**, as soon as you have read the information on the page. **Response:** The dialog **Communication Protocol** is opened.

Define SA85 adapter

Carry out the following steps to define the LAN address set in the CONFIG.SYS file:

Step	Action
1	Select the option button **Modbus Plus**.
2	Click on the command button **Next**. **Response:** The **Modbus Plus Target** dialog is opened.
3	Select from the list **Devices Online:** the adapter address you set when installing the SA85 in the CONFIG.SYS file (Parameter /n).

Protocol settings

Carry out the following steps to set the Modbus Plus protocol settings:

Step	Action
1	Click on the command button **Scan**. **Response:** The nodes on the Modbus Plus network are read and displayed graphically in the left window. A green point in the graphic indicates that the CPU is in RUN mode. To stop the CPU continue as described in step 3.
2	Double-click on the read network node in the graphical display. **Response:** The Modbus address of the node is automatically entered in the textfeld In the text field **Modbus Plus Address**.
3	Click the right mouse button in the left window. **Response:** A context menu with individual PLC commands is opened.
4	If the "Run" display is lit on the CPU, stop the program using the command **Stop PLC**. **Response:** A message window appears where you can click **OK** to confirm stopping the CPU.
5	Click the command button **OK**, to confirm stopping the CPU. **Response:** You return to the **Modbus Plus Target** dialog, and the green point disappears from the graph.
6	Activate the check box **Bridge**, if the connection to the Modbus network node should be made via a Modbus network using a Modbus bridge.
7	Press the appropriate Option button for your system (**PLC**, **Local Head**, **Remote I/O Drop**).
8	Click on the command button **Next**. **Response:** The **Operation** dialog is opened.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Press the Option button **Transfer EXEC to Device**.
2	Click on the command button **Browse...**. **Response:** The Concept directory is opened in a standard window.
3	Double-click on the DAT directory. **Response:** All available*.BIN files are displayed.
4	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Atrium PLC Types, page 1104*. **Response:** The selected *.BIN file is displayed in **File name:** text field.
5	Click on the command button **Open**. **Response:** You return to the dialog **Operation**, and the path to the selected *.BIN file is displayed in the **Filename** text field.
6	Click on the command button **Next**. **Response:** The dialog **File and Device Info** is opened. Information is provided here about the selected *.BIN file and also about the PLC.
7	Click on the command button **Next**. **Response:** The **Summary** dialog is opened. This gives you an overview of the settings made for you to check.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Click on the command button **Transfer**. **Response:** A message box appears warning you that all data available on the PLC will be lost, and the configuration and program must be reloaded on the PLC.
2	Click on the command button **Yes**, to continue the transfer. **Response:** The **Progress** dialog is opened. This gives information about the progress of the transfer in a progress bar and text.
3	Click **Close** once the transfer is complete. **Response:** The dialog is closed, and you return to the dialog **Summary**.
4	Click on the command button **Close**, to close the EXECLoader.

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

NOTE: If you recieve an error message, close Concept and start the BDRESET.EXE file (in the \Concept directory) to reset the SA85. Then start again from the first step.

Momentum first startup for IEC with EXECLoader

Introduction

This section describes the first startup of Momentum for IEC when used with Modbus Plus.

The first startup is subdivided into 5 main sections:

- *Start EXECLoader, page 1076*
- *Define SA85 adapter, page 1076*
- *Protocol settings, page 1077*
- *Select EXEC file, page 1078*
- *Load EXEC file, page 1078*

Start EXECLoader

The procedure for launching EXECLoaders is as follows:

Step	Action
1	Open the Concept start menu. **Response:** All installed Concept programs are displayed as symbols.
2	Click on the symbol with the identifier **EXECLoader**. **Response:** The EXECLoader program is started.
3	Click on the command **Next**, as soon as you have read the information on the page. **Response:** The dialog **Communication Protocol** is opened.

Define SA85 adapter

Carry out the following steps to define the LAN address set in the CONFIG.SYS file:

Step	Action
1	Select the option button **Modbus Plus**.
2	Click on the command button **Next**. **Response:** The **Modbus Plus Target** dialog is opened.
3	Select from the list **Devices Online:** the adapter address you set when installing the SA85 in the CONFIG.SYS file (Parameter /n).

Protocol settings

Carry out the following steps to set the Modbus Plus protocol settings:

Step	Action
1	Click on the command button **Scan**. **Response:** The nodes on the Modbus Plus network are read and displayed graphically in the left window. A green point in the graphic indicates that the CPU is in RUN mode. To stop the CPU continue as described in step 3.
2	Double-click on the read network node in the graphical display. **Response:** The Modbus address of the node is automatically entered in the textfeld In the text field **Modbus Plus Address**.
3	Click the right mouse button in the left window. **Response:** A context menu with individual PLC commands is opened.
4	If the "Run" display is lit on the CPU, stop the program using the command **Stop PLC**. **Response:** A message window appears where you can click **OK** to confirm stopping the CPU.
5	Click the command button **OK**, to confirm stopping the CPU. **Response:** You return to the **Modbus Plus Target** dialog, and the green point disappears from the graph.
6	Activate the check box **Bridge**, if the connection to the Modbus network node should be made via a Modbus network using a Modbus bridge.
7	Press the appropriate Option button for your system (**PLC**, **Local Head**, **Remote I/O Drop**).
8	Click on the command button **Next**. **Response:** The **Operation** dialog is opened.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Press the Option button **Transfer EXEC to Device**.
2	Click on the command button **Browse...**. **Response:** The Concept directory is opened in a standard window.
3	Double-click on the DAT directory. **Response:** All available*.BIN files are displayed.
4	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Momentum PLC Types, page 1103*. **Response:** The selected *.BIN file is displayed in **File name:** text field.
5	Click on the command button **Open**. **Response:** You return to the dialog **Operation**, and the path to the selected *.BIN file is displayed in the **Filename** text field.
6	Click on the command button **Next**. **Response:** The dialog **File and Device Info** is opened. Information is provided here about the selected *.BIN file and also about the PLC.
7	Click on the command button **Next**. **Response:** The **Summary** dialog is opened. This gives you an overview of the settings made for you to check.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Click on the command button **Transfer**. **Response:** A message box appears warning you that all data available on the PLC will be lost, and the configuration and program must be reloaded on the PLC.
2	Click on the command button **Yes**, to continue the transfer. **Response:** The **Progress** dialog is opened. This gives information about the progress of the transfer in a progress bar and text.
3	Click **Close** once the transfer is complete. **Response:** The dialog is closed, and you return to the dialog **Summary**.
4	Click on the command button **Close**, to close the EXECLoader.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Run	slow flashing
COM ACT	not lit
Modbus +	fast flashing

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Run	not lit
COM ACT	not lit
Modbus +	fast flashing

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

NOTE: If you recieve an error message, close Concept and start the BDRESET.EXE file (in the \Concept directory) to reset the SA85. Then start again from the first step.

Momentum first startup for LL984 with EXECLoader

Introduction

This section describes the first startup of Momentum for LL984 when used with Modbus Plus.

NOTE: Loading the EXEC file for LL984 is not necessary with a new computer, since it is preloaded in the the CPUs Flash RAM. Loading the EXEC file for LL984 is only necessary if you have already loaded the EXEC file for IEC, and now wish to change.

You should always check to see if a new EXEC version has been released in the meantime. This information and the current EXEC file can be found on our website at www.schneiderautomation.com. You can see the currently loaded version of the EXEC file in Concept using the **Online →Online controller...** menu command.

The first startup is subdivided into 5 main sections:

- *Start EXECLoader, page 1080*
- *Define SA85 adapter, page 1080*
- *Protocol settings, page 1081*
- *Select EXEC file, page 1082*
- *Load EXEC file, page 1082*

Start EXECLoader

The procedure for launching EXECLoaders is as follows:

Step	Action
1	Open the Concept start menu. **Response:** All installed Concept programs are displayed as symbols.
2	Click on the symbol with the identifier **EXECLoader**. **Response:** The EXECLoader program is started.
3	Click on the command **Next**, as soon as you have read the information on the page. **Response:** The dialog **Communication Protocol** is opened.

Define SA85 adapter

Carry out the following steps to define the LAN address set in the CONFIG.SYS file:

Step	Action
1	Select the option button **Modbus Plus**.
2	Click on the command button **Next**. **Response:** The **Modbus Plus Target** dialog is opened.
3	Select from the list **Devices Online:** the adapter address you set when installing the SA85 in the CONFIG.SYS file (Parameter /n).

Protocol settings

Carry out the following steps to set the Modbus Plus protocol settings:

Step	Action
1	Click on the command button **Scan**. **Response:** The nodes on the Modbus Plus network are read and displayed graphically in the left window. A green point in the graphic indicates that the CPU is in RUN mode. To stop the CPU continue as described in step 3.
2	Double-click on the read network node in the graphical display. **Response:** The Modbus address of the node is automatically entered in the textfeld In the text field **Modbus Plus Address**.
3	Click the right mouse button in the left window. **Response:** A context menu with individual PLC commands is opened.
4	If the "Run" display is lit on the CPU, stop the program using the command **Stop PLC**. **Response:** A message window appears where you can click **OK** to confirm stopping the CPU.
5	Click the command button **OK**, to confirm stopping the CPU. **Response:** You return to the **Modbus Plus Target** dialog, and the green point disappears from the graph.
6	Activate the check box **Bridge**, if the connection to the Modbus network node should be made via a Modbus network using a Modbus bridge.
7	Press the appropriate Option button for your system (**PLC**, **Local Head**, **Remote I/O Drop**).
8	Click on the command button **Next**. **Response:** The **Operation** dialog is opened.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Press the Option button **Transfer EXEC to Device**.
2	Click on the command button **Browse...**. **Response:** The Concept directory is opened in a standard window.
3	Double-click on the DAT directory. **Response:** All available*.BIN files are displayed.
4	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Momentum PLC Types, page 1103*. **Response:** The selected *.BIN file is displayed in **File name:** text field.
5	Click on the command button **Open**. **Response:** You return to the dialog **Operation**, and the path to the selected *.BIN file is displayed in the **Filename** text field.
6	Click on the command button **Next**. **Response:** The dialog **File and Device Info** is opened. Information is provided here about the selected *.BIN file and also about the PLC.
7	Click on the command button **Next**. **Response:** The **Summary** dialog is opened. This gives you an overview of the settings made for you to check.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Click on the command button **Transfer**. **Response:** A message box appears warning you that all data available on the PLC will be lost, and the configuration and program must be reloaded on the PLC.
2	Click on the command button **Yes**, to continue the transfer. **Response:** The **Progress** dialog is opened. This gives information about the progress of the transfer in a progress bar and text.
3	Click **Close** once the transfer is complete. **Response:** The dialog is closed, and you return to the dialog **Summary**.
4	Click on the command button **Close**, to close the EXECLoader.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Run	slow flashing
COM ACT	not lit
Modbus +	fast flashing

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Run	not lit
COM ACT	not lit
Modbus +	fast flashing

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

NOTE: If you recieve an error message, close Concept and start the BDRESET.EXE file (in the \Concept directory) to reset the SA85. Then start again from the first step.

Startup when using Modbus Plus with DOS Loader

O

Overview

This chapter describes loading executive data (EXEC) onto the PLC with the DOS Loader program.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Quantum first startup with DOS Loader	1086
Compact first startup with DOS Loader	1089
Atrium first startup with DOS Loader	1092
Momentum first startup for IEC with DOS Loader	1095
Momentum first startup for LL984 with DOS Loader	1098

Quantum first startup with DOS Loader

Introduction

This section describes the first startup of Quantum when used with Modbus Plus.

The first startup is subdivided into 5 main sections:
- *Start DOS Loader, page 1086*
- *Define SA85 adapter, page 1086*
- *Protocol settings, page 1087*
- *Select EXEC file, page 1087*
- *Load EXEC file, page 1087*

Start DOS Loader

The procedure for launching DOS Loaders is as follows:

Step	Action
1	If the CPU display "Run" is lit, stop the program with Concept (in main menu **Online**). **Response:** The "Run" display is no longer lit; the "Ready" is now lit.
2	Open the directory DAT (CONCEPT\DAT).
3	Double-click on the file LOADER.EXE. **Response:** The installation program for the Executive file (EXEC) is started.
4	Select the option **Communication Parameters**. **Response:** The dialog box **Communication setup** is opened.
5	Select the **Modbus Plus**option. **Response:** The **Modbus Plus communication setup** window is opened.

Define SA85 adapter

Carry out the following steps to define the LAN address set in the CONFIG.SYS file:

Step	Action
1	Select the adapter address you set when installing the SA85 in the CONFIG.SYS file (Parameter /n).
2	Confirm your selection with RETURN.

Protocol settings

Carry out the following steps to set the Modbus Plus protocol settings:

Step	Action
1	Enter in **Enter First Routing Path:** the node address set on the CPU (using a rotary switch on the back of the module).
2	Confirm the entry with RETURN. **Response:** The option **Enter second Routing Path:** appears.
3	Acknowledge the option with RETURN. **Response:** The window for selecting the TARGET PATH appears.
4	Select the **TARGET PATH 1**option.
5	Enter in **Enter Software Interrupt ->** the Interrupt (5c or 5d) selected in the CONFIG.SYS file.
6	Select the **ACCEPT CHANGES**option. **Response:** You return to the main menu.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Select the option **File Selection**. **Response:** The **File Selection** window is opened.
2	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Quantum PLC Types, page 1101*.
3	Confirm your selection with RETURN. **Response:** You return to the main menu.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Select the option **Load File To PLC**. **Response:** The **Loading Process** window is opened, and the warning "The PROGRAM contents of the device being loaded could be lost after loading a new Executive. The CONTENTS stored in the Micro H H P will always be lost after loading a new Executive. Would you like to continue (Y/N) ? N" is displayed.
2	Acknowledge the warning with **Y**. **Response:** The message "Node failed to enter normal mode" appears.
3	The DOWNLOAD PROGRESS window appears which shows the transfer rate. **Response:** After the transfer is complete, the message "Download Operation Successful" appears.

Step	Action
4	Confirm the message with RETURN. **Response:** You return to the main menu.
5	Select the **Exit Program**option. **Response:** The DOS Loader is exited.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	slow flashing
Modbus	not lit
Modbus +	fast flashing

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Ready	lit
Run	not lit
Modbus	not lit
Modbus +	fast flashing

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

NOTE: If you recieve an error message, close Concept and start the BDRESET.EXE file (in the \Concept directory) to reset the SA85. Then start again from the first step.

Compact first startup with DOS Loader

Introduction

This section describes the first startup of Compact when used with Modbus Plus.

The first startup is subdivided into 5 main sections:
- *Start DOS Loader, page 1089*
- *Define SA85 adapter, page 1089*
- *Protocol settings, page 1090*
- *Select EXEC file, page 1090*
- *Load EXEC file, page 1091*

Start DOS Loader

The procedure for launching DOS Loaders is as follows:

Step	Action
1	If the CPU display "Run" is lit, stop the program with Concept (in main menu **Online**). **Response:** The "Run" display is no longer lit; the "Ready" is now lit.
2	Open the directory DAT (CONCEPT\DAT).
3	Double-click on the file LOADER.EXE. **Response:** The installation program for the Executive file (EXEC) is started.
4	Select the option **Communication Parameters**. **Response:** The dialog box **Communication setup** is opened.
5	Select the **Modbus Plus**option. **Response:** The **Modbus Plus communication setup** window is opened.

Define SA85 adapter

Carry out the following steps to define the LAN address set in the CONFIG.SYS file:

Step	Action
1	Select the adapter address you set when installing the SA85 in the CONFIG.SYS file (Parameter /n).
2	Confirm your selection with RETURN.

Protocol settings

Carry out the following steps to set the Modbus Plus protocol settings:

Step	Action
1	Enter in **Enter First Routing Path:** the node address set on the CPU (using a rotary switch on the back of the module).
2	Confirm the entry with RETURN. **Response:** The option **Enter second Routing Path:** appears.
3	Acknowledge the option with RETURN. **Response:** The window for selecting the TARGET PATH appears.
4	Select the **TARGET PATH 1**option.
5	Enter in **Enter Software Interrupt ->** the Interrupt (5c or 5d) selected in the CONFIG.SYS file.
6	Select the **ACCEPT CHANGES**option. **Response:** You return to the main menu.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Select the option **File Selection**. **Response:** The **File Selection** window is opened.
2	Select the *.BIN file CTSX201D.
3	Confirm your selection with RETURN. **Response:** You return to the main menu.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Select the option **Load File To PLC**. **Response:** The **Loading Process** window is opened, and the warning "The PROGRAM contents of the device being loaded could be lost after loading a new Executive. The CONTENTS stored in the Micro H H P will always be lost after loading a new Executive. Would you like to continue (Y/N) ? N" is displayed.
2	Acknowledge the warning with **Y**. **Response:** The message "Node failed to enter normal mode" appears.
3	The DOWNLOAD PROGRESS window appears which shows the transfer rate. **Response:** After the transfer is complete, the message "Download Operation Successful" appears.
4	Confirm the message with RETURN. **Response:** You return to the main menu.
5	Select the **Exit Program**option. **Response:** The DOS Loader is exited.

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

NOTE: If you recieve an error message, close Concept and start the BDRESET.EXE file (in the \Concept directory) to reset the SA85. Then start again from the first step.

Atrium first startup with DOS Loader

Introduction

This section describes the first startup of Atrium when used with Modbus Plus. The hardware requirements for loading EXEC files can be seen in the "Modicon TSX Atrium" manual.

The first startup is subdivided into 5 main sections:

- *Start DOS Loader, page 1092*
- *Define SA85 adapter, page 1092*
- *Protocol settings, page 1093*
- *Select EXEC file, page 1093*
- *Load EXEC file, page 1094*

Start DOS Loader

The procedure for launching DOS Loaders is as follows:

Step	Action
1	If the CPU display "Run" is lit, stop the program with Concept (in main menu **Online**). **Response:** The "Run" display is no longer lit; the "Ready" is now lit.
2	Open the directory DAT (CONCEPT\DAT).
3	Double-click on the file LOADER.EXE. **Response:** The installation program for the Executive file (EXEC) is started.
4	Select the option **Communication Parameters**. **Response:** The dialog box **Communication setup** is opened.
5	Select the **Modbus Plus**option. **Response:** The **Modbus Plus communication setup** window is opened.

Define SA85 adapter

Carry out the following steps to define the LAN address set in the CONFIG.SYS file:

Step	Action
1	Select the adapter address you set when installing the SA85 in the CONFIG.SYS file (Parameter /n).
2	Confirm your selection with RETURN.

Protocol settings

Carry out the following steps to set the Modbus Plus protocol settings:

Step	Action
1	Enter in **Enter First Routing Path:** the node address set on the CPU (using a rotary switch on the back of the module).
2	Confirm the entry with RETURN. **Response:** The option **Enter second Routing Path:** appears.
3	Acknowledge the option with RETURN. **Response:** The window for selecting the TARGET PATH appears.
4	Select the **TARGET PATH 1**option.
5	Enter in **Enter Software Interrupt ->** the Interrupt (5c or 5d) selected in the CONFIG.SYS file.
6	Select the **ACCEPT CHANGES**option. **Response:** You return to the main menu.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Select the option **File Selection**. **Response:** The **File Selection** window is opened.
2	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Atrium PLC Types, page 1104*.
3	Confirm your selection with RETURN. **Response:** You return to the main menu.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Select the option **Load File To PLC**. **Response:** The **Loading Process** window is opened, and the warning "The PROGRAM contents of the device being loaded could be lost after loading a new Executive. The CONTENTS stored in the Micro H H P will always be lost after loading a new Executive. Would you like to continue (Y/N) ? N" is displayed.
2	Acknowledge the warning with **Y**. **Response:** The message "Node failed to enter normal mode" appears.
3	The DOWNLOAD PROGRESS window appears which shows the transfer rate. **Response:** After the transfer is complete, the message "Download Operation Successful" appears.
4	Confirm the message with RETURN. **Response:** You return to the main menu.
5	Select the **Exit Program**option. **Response:** The DOS Loader is exited.

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

NOTE: If you recieve an error message, close Concept and start the BDRESET.EXE file (in the \Concept directory) to reset the SA85. Then start again from the first step.

Momentum first startup for IEC with DOS Loader

Introduction

This section describes the first startup of Momentum for IEC when used with Modbus Plus.

The first startup is subdivided into 5 main sections:

- *Start DOS Loader, page 1095*
- *Define SA85 adapter, page 1095*
- *Protocol settings, page 1096*
- *Select EXEC file, page 1096*
- *Load EXEC file, page 1096*

Start DOS Loader

The procedure for launching DOS Loaders is as follows:

Step	Action
1	If the CPU display "Run" is lit, stop the program with Concept (in main menu **Online**). **Response:** The "Run" display is no longer lit; the "Ready" is now lit.
2	Open the directory DAT (CONCEPT\DAT).
3	Double-click on the file LOADER.EXE. **Response:** The installation program for the Executive file (EXEC) is started.
4	Select the option **Communication Parameters**. **Response:** The dialog box **Communication setup** is opened.
5	Select the **Modbus Plus**option. **Response:** The **Modbus Plus communication setup** window is opened.

Define SA85 adapter

Carry out the following steps to define the LAN address set in the CONFIG.SYS file:

Step	Action
1	Select the adapter address you set when installing the SA85 in the CONFIG.SYS file (Parameter /n).
2	Confirm your selection with RETURN.

Protocol settings

Carry out the following steps to set the Modbus Plus protocol settings:

Step	Action
1	Enter in **Enter First Routing Path:** the node address set on the CPU (using a rotary switch on the back of the module).
2	Confirm the entry with RETURN. **Response:** The option **Enter second Routing Path:** appears.
3	Acknowledge the option with RETURN. **Response:** The window for selecting the TARGET PATH appears.
4	Select the **TARGET PATH 1**option.
5	Enter in **Enter Software Interrupt ->** the Interrupt (5c or 5d) selected in the CONFIG.SYS file.
6	Select the **ACCEPT CHANGES**option. **Response:** You return to the main menu.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Select the option **File Selection**. **Response:** The **File Selection** window is opened.
2	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Momentum PLC Types, page 1103*.
3	Confirm your selection with RETURN. **Response:** You return to the main menu.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Select the option **Load File To PLC**. **Response:** The **Loading Process** window is opened, and the warning "The PROGRAM contents of the device being loaded could be lost after loading a new Executive. The CONTENTS stored in the Micro H H P will always be lost after loading a new Executive. Would you like to continue (Y/N) ? N" is displayed.
2	Acknowledge the warning with **Y**. **Response:** The message "Node failed to enter normal mode" appears.
3	The DOWNLOAD PROGRESS window appears which shows the transfer rate. **Response:** After the transfer is complete, the message "Download Operation Successful" appears.

Step	Action
4	Confirm the message with RETURN. **Response:** You return to the main menu.
5	Select the **Exit Program**option. **Response:** The DOS Loader is exited.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Run	slow flashing
COM ACT	not lit
Modbus +	fast flashing

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Run	not lit
COM ACT	not lit
Modbus +	fast flashing

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

NOTE: If you recieve an error message, close Concept and start the BDRESET.EXE file (in the \Concept directory) to reset the SA85. Then start again from the first step.

Momentum first startup for LL984 with DOS Loader

Introduction

This section describes the first startup of Momentum for LL984 when used with Modbus Plus.

NOTE: Loading the EXEC file for LL984 is not necessary with a new computer, since it is preloaded in the the CPUs Flash RAM. Loading the EXEC file for LL984 is only necessary if you have already loaded the EXEC file for IEC, and now wish to change.

You should always check to see if a new EXEC version has been released in the meantime. This information and the current EXEC file can be found on our website at www.schneiderautomation.com. You can see the currently loaded version of the EXEC file in Concept using the **Online →Online controller...** menu command.

The first startup is subdivided into 5 main sections:

- *Start DOS Loader, page 1098*
- *Define SA85 adapter, page 1099*
- *Protocol settings, page 1099*
- *Select EXEC file, page 1099*
- *Load EXEC file, page 1100*

Start DOS Loader

The procedure for launching DOS Loaders is as follows:

Step	Action
1	If the CPU display "Run" is lit, stop the program with Concept (in main menu **Online**). **Response:** The "Run" display is no longer lit; the "Ready" is now lit.
2	Open the directory DAT (CONCEPT\DAT).
3	Double-click on the file LOADER.EXE. **Response:** The installation program for the Executive file (EXEC) is started.
4	Select the option **Communication Parameters**. **Response:** The dialog box **Communication setup** is opened.
5	Select the **Modbus Plus**option. **Response:** The **Modbus Plus communication setup** window is opened.

Define SA85 adapter

Carry out the following steps to define the LAN address set in the CONFIG.SYS file:

Step	Action
1	Select the adapter address you set when installing the SA85 in the CONFIG.SYS file (Parameter /n).
2	Confirm your selection with RETURN.

Protocol settings

Carry out the following steps to set the Modbus Plus protocol settings:

Step	Action
1	Enter in **Enter First Routing Path:** the node address set on the CPU (using a rotary switch on the back of the module).
2	Confirm the entry with RETURN. **Response:** The option **Enter second Routing Path:** appears.
3	Acknowledge the option with RETURN. **Response:** The window for selecting the TARGET PATH appears.
4	Select the **TARGET PATH 1**option.
5	Enter in **Enter Software Interrupt ->** the Interrupt (5c or 5d) selected in the CONFIG.SYS file.
6	Select the **ACCEPT CHANGES**option. **Response:** You return to the main menu.

Select EXEC file

Carry out the following steps to select the EXEC file:

Step	Action
1	Select the option **File Selection**. **Response:** The **File Selection** window is opened.
2	Click on the *.BIN file that corresponds to your CPU and the desired programming language. See the table *Loading Firmware for Momentum PLC Types, page 1103*.
3	Confirm your selection with RETURN. **Response:** You return to the main menu.

Load EXEC file

Carry out the following steps to load the EXEC file in the CPU flash RAM:

Step	Action
1	Select the option **Load File To PLC**. **Response:** The **Loading Process** window is opened, and the warning "The PROGRAM contents of the device being loaded could be lost after loading a new Executive. The CONTENTS stored in the Micro H H P will always be lost after loading a new Executive. Would you like to continue (Y/N) ? N" is displayed.
2	Acknowledge the warning with **Y**. **Response:** The message "Node failed to enter normal mode" appears.
3	The DOWNLOAD PROGRESS window appears which shows the transfer rate. **Response:** After the transfer is complete, the message "Download Operation Successful" appears.
4	Confirm the message with RETURN. **Response:** You return to the main menu.
5	Select the **Exit Program**option. **Response:** The DOS Loader is exited.

CPU display during transfer

During transfer the CPU display is as follows:

LED	Response
Run	slow flashing
COM ACT	not lit
Modbus +	fast flashing

CPU display after transfer

After transfer the CPU display is as follows:

LED	Response
Run	not lit
COM ACT	not lit
Modbus +	fast flashing

Creating the software connection

Carry out the steps given in chapter *Creating a Project, page 75*.

NOTE: If you recieve an error message, close Concept and start the BDRESET.EXE file (in the \Concept directory) to reset the SA85. Then start again from the first step.

EXEC files

P

Loading Firmware

At a Glance

You obtain the PLC types of the different firmware by loading the EXEC files (*.BIN).

Loading Firmware for Quantum PLC Types

Assigning the EXEC files:

140 CPU	Q186Vxxx (IEC+LL984)	Q486Vxxx (IEC+LL984)	Q58VxxxE (IEC+LL984)	Q5RVxxxE (IEC+LL984)	Q1SVxxxE (IEC only) *	IEC memory (kbyte)
113 02	X (LL984 only)	-	-	-	-	
113 02S	-	-	-	-	X	max. 120
113 02X	X (LL984 only)	-	-	-	-	
113 03	X	-	-	-	-	max. 138
113 03S	-	-	-	-	X	max. 380
113 03X	X	-	-	-	-	max. 136
213 04	X	-	-	-	-	max. 320
213 04S	-	-	-	-	X	max. 604
213 04X	X	-	-	-	-	max. 320
424 0x	-	X	-	-	-	max. 307
424 0xX	-	X	-	-	-	max. 307
434 12	-	-	X	-	-	max. 892
534 14	-	-	X	-	-	max. 2556

140 CPU	Q186Vxxx (IEC+LL984)	Q486Vxxx (IEC+LL984)	Q58VxxxE (IEC+LL984)	Q5RVxxxE (IEC+LL984)	Q1SVxxxE (IEC only) *	IEC memory (kbyte)
434 12A (Redesigned CPU)	-	-	-	X	-	max. 892
534 14A/B (Redesigned CPU	-	-	-	X	-	max. 2556

NOTE: * After the Q1SVxxxE.BIN EXEC file is loaded the EMUQ.EXE loadable must be loaded in Concept in the **Loadables** (**PLC Configuration** →**Loadables...**) dialog box.

Loading Firmware for Quantum LL984 Hot Standby Operation

The Quantum CPUs not ending in X or S can be used for the LL984 Hot Standby mode. A special EXEC file must be downloaded onto the CPU for this. The loadable for LL984 Hot Standby (CHS_208.DAT) is automatically installed by the system.

Loading Firmware for Quantum IEC Hot Standby Operation

The CPUs 140 CPU 434 12 and 140 CPU 534 14 can also be used for IEC Hot Standby. A special EXEC file must be downloaded onto the CPU for this. The loadables for IEC Hot Standby (IHSB196.EXE and CHS_208.DAT) are automatically installed by the system.

Loading Firmware for Quantum Equation Editor

The Quantum CPUs not ending in X or S can be used for the LL984 equation editor. A special EXEC file must be downloaded onto the CPU flash for this. This EXEC file is not part of the Concept delivery range but can be obtained over the Internet at www.schneiderautomation.com.

Loading Firmware for Momentum PLC Types

Momentum PLC type (CPU 171 CBB 970 30):

171 CBB	MPSV100.BIN (LL984 only)	MPSV100e.BIN (IEC only)	IEC memory (kbyte)
970 30-984	X	-	
970 30-IEC	-	X	236

Assigning the EXEC files for Momentum PLC type (CPU 171 CCC 7x0 x0):

171 CCC	M1LLVxxx (LL984 only)	M1IVxxxE (IEC only)	IEC memory (kbyte)
760 10-984	X	-	
760 10-IEC	-	X	220
780 10-984	X	-	
780 10-IEC	-	X	220

Assigning the EXEC files for Momentum PLC type (CPU 171 CCC 9x0 x0):

171 CCC	M1EVxxx (LL984 only)	M1EVxxxE (IEC only)	IEC memory (kbyte)
960 20-984	X	-	
960 30-984	X	-	
960 30-IEC	-	X	236
980 20-984	X	-	
980 30-984	X	-	
980 30-IEC	-	X	236

Assigning the EXEC files for Momentum PLC type (CPU 171 CCS 7x0 x0):

171 CCS	M1LLVxxx (LL984 only)	M1IVxxxE (IEC only)	IEC memory (kbyte)
700 10	X	-	
700/780 00	X	-	
760 00-984	X	-	
760 00-IEC	-	X	160

The stripped EXEC of the M1 supports up to a maximum of 44 I/O modules.

Loading Firmware for Compact PLC Types

The **CTSXxxxD.BIN** EXEC file must be downloaded onto the CPU flash for all Compact CPUs.

Loading Firmware for Atrium PLC Types

A special EXEC file must be downloaded onto the CPU flash for each Atrium CPU (see table below).

180 CCO	EXEC file
121 01	AI3Vxxx.BIN
241 01	AI5Vxxx.BIN
241 11	AI5Vxxx.BIN

INI Files

Q

Overview

This chapter contains settings that can be made in several INI files.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
Q.1	Settings in the CONCEPT.INI File	1106
Q.2	Settings in the Projectname.INI File	1120

Q.1 Settings in the CONCEPT.INI File

Overview

This section describes the settings in the CONCEPT.INI file.

What's in this Section?

This section contains the following topics:

Topic	Page
General information on the Concept INI file	1107
INI Print Settings	1108
INI Settings for Register Address Format, Variable Storage and Project Name Definition	1109
INI-settings for path information and global DFBs [Path] [Upload]	1110
Representation of Internal Data in the INI File	1113
INI Settings for the LD Section	1114
INI Settings for Online Processing [Colors]	1115
INI Settings for Warning Messages and the Address Format	1116
INI Security Settings	1117
INI-Settings for the RDE behavior	1118
INI settings for the Options> Toolsmenu	1119

General information on the Concept INI file

Introduction

Software settings can be specified in the Concept INI file. Settings generated by the program are also stored in the INI file. The INI file initially contains defaults that can subsequently be changed.

Where is the CONCEPT.INI file situated?

After the installation of Concept, the CONCEPT.INI file can be found in the Windows directory.

Editing the INI File

Various settings are given (exception: path information) and divided into several keywords. The lines that begin with a semicolon (;) represent comments or explain the subsequent setting.

To edit the INI file, only change the lines without semicolons (;) or insert a new line after the comment, in which to specify the path. Then save the modified file.

NOTE: Changes in the INI file are only accepted after Concept/DFB Editor/Converter is restarted.

INI Print Settings

Printing FBD Sections

Defining default page break values for FBD sections:

Setting	Description
`DX_FBD_PORTRAIT=`	Specify portrait width (default value at delivery = 75)
`DY_FBD_PORTRAIT=`	Specify portrait height (default value at delivery = 100)
`DX_FBD_LANDSCAPE=`	Specify landscape width (default value at delivery = 100)
`DY_FBD_LANDSCAPE=`	Specify landscape height (default value at delivery = 50)

Printing LD Sections

Defining default page break values for LD sections:

Setting	Description
`DX_LD_PORTRAIT=`	Specify portrait width (default value at delivery = 70)
`DY_LD_PORTRAIT=`	Specify portrait height (default value at delivery = 35)
`DX_LD_LANDSCAPE=`	Specify landscape width (default value at delivery = 105)
`DY_LD_LANDSCAPE=`	Specify landscape height (default value at delivery = 18)

Printing SFC Sections

Defining default page break values for SFC sections:

Setting	Description
`DX_SFC_PORTRAIT=`	Specify portrait width (default value at delivery = 11)
`DY_SFC_PORTRAIT=`	Specify portrait height (default value at delivery = 20)
`DX_SFC_LANDSCAPE=`	Specify landscape width (default value at delivery = 15)
`DY_SFC_LANDSCAPE=`	Specify landscape height (default value at delivery = 11)

INI Settings for Register Address Format, Variable Storage and Project Name Definition

Defining the register address format [Common]

Specifying the register address format (e.g. 4x reference):

Setting	Description
`AddrStyle=0`	0 = 400001 (default)
`AddrStyle=1`	1 = 4:00001 (separator)
`AddrStyle=2`	2 = 4:1 (compact)
`AddrStyle=3`	3 = QW00001 (IEC)

Defining variable storage [Common]

Store variables in file:

Setting	Description
`ExportVariables=1`	After a project has been downloaded and saved, all variables are stored in a file. This file is called *.VAR and is found in the **"Project directory"** →**VAR** →***.VAR**. All variables and their attributes are shown in this file.
`ExportVariables=0`	Additional storage of variables in a file does not take place.

Determining the validity of digits in project names [Common]

Determining the validity of digits in project names:

Setting	Description
`ProjectPrefixDigit=1`	Project names that begin with a digit are allowed.
`ProjectPrefixDigit=0`	Project names that begin with a digit are not allowed.

Determining the validity of located variables in DFBs [Common]

Determining the validity of located variables in DFBs:

Setting	Description
`AllowLocatedVarsInDFB=1`	Located variables are allowed in DFBs. **Note:** This setting can also be made directly in Concept in **Options** →**Preferences** →**IEC Extensions** → **Allow Located Variable in DFBs** dialog box.

INI-settings for path information and global DFBs [Path] [Upload]

Defining the Path for Global DFBs and Help Files [Path]

Defining paths:

Setting	Description
`GlobalDFBPath=`	Specify path for global DFBs.
`HelpPath=`	Specify path for help files.

Defining the Storage of Global DFBs during Upload

Defining a new directory for global DFBs:

Setting	Description
`PreserveGlobalDFBs=1`	During the upload process, a GLB directory for the global DFBs is created in the project directory. By doing this, existing global DFBs in the Concept DFB directory will not be overwritten. **Advantage:** No impact on other projects, as the global DFBs in these projects are not overwritten. **Disadvantage:** Multiple copies of global DFBs. **Note:** Also read the sections entitled *How are Global DFBs Stored?, page 1111* and *How are Global DFBs Read?, page 1112*.
`PreserveGlobalDFBs=0`	During the upload process, global DFBs are downloaded into the Concept DFB directory. Different versions of duplicated DFBs are recognized and overwritten after being queried. **Advantage:** Only one copy of global DFBs for several projects. **Disadvantage:** Existing global DFBs whose versions differ from the uploaded DFBs are overwritten. This can cause other projects to be inconsistent in certain circumstances. **Note:** Also read the sections entitled *How are Global DFBs Stored?, page 1111* and *How are Global DFBs Read?, page 1112*.

Define exclusion of global/local DFBs from Online-Backup [Backup]

Define exclusion of global and/or local DFBs from Online-Backup :

Setting	Description
`ExcludeAllDFBs=1`	All DFBs are excluded from Online-Backup. I.e. for the **Online →Loading** or **Online →Load changes** the backup-directory does not contain the "DFB" and "DFB.GLB". The standard setting contains no entries, i.e. all DFBs are present in the Backup directory. **Note**: This setting is applied to all projects on the PC.
`ExcludeGlobalDFBs=1`	Global DFBs are excluded from Online-Backup. I.e. for the **Online →Loading** or **Online →Load changes** the backup-directory does not contain the directory "DFB.GLB". The standard setting contains no entries, i.e. all DFBs are present in the Backup directory. **Note**: This setting is applied to all projects on the PC.

How are Global DFBs Stored?

Storage of global DFBs depends on the settings in the INI file:

If a project...	then the global DFBs are...
is recreated, and no new DFB path has been defined in the INI file,	stored in the x:\CONCEPT\DFB directory.
is recreated, and a DFB path has been defined in the INI file,	stored in the DFB directory of the defined path.
is uploaded, and the following settings exist in the INI file: - the [Path] option "GlobalDFBPath=x:\DFB", - the [Upload] option "PreserveGlobalDFBs=0",	stored in the DFB directory defined in the path (x:\DFB).
is uploaded, and the following settings exist in the INI file: - the [Path] option "GlobalDFBPath=x:\DFB", - the [Upload] option "PreserveGlobalDFBs=1",	stored in the project's GLB directory. **Note:** The GLB directory is always used first, as soon as the "PreserveGlobalDFBs=1" [Upload] option is specified.

How are Global DFBs Read?

When a project is opened, the system looks for global DFBs in the following order:

Step	Description
1	The project directory is searched for an existing GLB directory.
2	The relevant settings are checked in the INI file. For example: [Path]: GlobalDFBPath=x:\DFB [Upload]: PreserveGlobalDFBs=0 In this example, the DFB directory of the path defined is searched for global DFBs.
3	The DFB directory in x:\CONCEPT\DFB is searched.

Only the global DFBs from one directory are used, i.e. if step 1 is unsuccessful, then step 2 follows, step 3 is only performed if neither of the first two are successful.

Representation of Internal Data in the INI File

Representation of Internal Data

The following keywords appear in the INI file and contain internal data according to specific Concept applications:

- [Debug]
- [Configurator]
- [Search]
- [Registration]
- [Register]

INI Settings for the LD Section

Defining the Contact Connection

Defining the contact connection to the power rail:

Setting	Description
`ExtendedAutoConnect=0`	Only the contacts from the first column in the LD editor are automatically connected to the power rail.
`ExtendedAutoConnect=1`	The contacts from the first and second columns in the LD editor are automatically connected to the power rail.

Defining the Number of Columns/Fields

Defining the number of columns/fields (only available when editing with keys):

Setting	Description
`AutowrapColumn=51`	The section contains 51 columns/fields by default. It is possible to set from 2 to 51 columns/fields. When the last column/field is reached, the following objects are automatically placed in the next lines. When this happens, a link with the previous lines is established, i.e. the objects are generated within a common rung. **Note:** Since with automatic line breaking, the objects that follow are placed in the second column/field, it is recommended that you set the contact connection to the power rail as `ExtendedAutoConnect=0`.

INI Settings for Online Processing [Colors]

Online Animation

Specify the representation of the line width and color:

Setting	Description
`AnimationSize=`	Specifying the line width of connections in FBD and LD and for objects in LD: The default setting is 1. It can be set from 1 to 10.
`ColorScheme=`	Specifying the color scheme for FBD, IL, ST, LD and SFC. It is possible to make a setting from 0 to 11. **Note:** An overview of the 12 different color schemes can be found in the online help (see "Colors" in the index).

INI Settings for Warning Messages and the Address Format

Multiple assignment [Warnings]

Reducing the number of warnings (referring to multiple assignment) in the message window:

Setting	Description
`Multiassignment=1`	Warning given if at least one variable X and a component X.C. was written.
`Multiassignment=0`	Warning given if one variable X was written at least twice as a whole.

Address format in LOG file [Logging]

Define address format in LOG file:

Setting	Description
`DD_MONTH_YYYY=1`	In Concept, the month is shown with 3 characters and in English. **Example:** 24-Dec-2002 14:46:24
`DD_MONTH_YYYY=0`	The format set in Windows is shown. The setting can be made in Windows with: **Control Panel** →**Regional Options** →**Date** →**Short date format:**

INI Security Settings

Concept Password Length [Securit]

Define the character length of the Concept password (see Concept Security):

Setting	Description
`MinPasswordLength=X`	The Concept password must have at least X characters. X = 6 to 12

INI-Settings for the RDE behavior

Uploaded State RAM in the RDE [RDE]

Define overwriting of the uploaded state RAM values

Setting	Description
`UpdateProjectStateRam=1`	Uploaded state RAM values can be overwritten in the RDE by Online-Operations (Standard Setting).
`UpdateProjectStateRam=0`	Uploaded state RAM values can not be overwritten in the RDE by Online-Operations.

RDE-Animation [RDE]

Define start of the RDE-Animation

Setting	Description
`StartWithAnimation=1`	RDE-Animation is automatically started when a table is opened.
`StartWithAnimation=0`	RDE-Animation is not started automatically when a table is opened. (Standard settings)

INI settings for the Options> Toolsmenu

Defining applications or help programs [TOOLS]

Defining applications or help programs to execute via the **Options** →**Tools** menu command:

Setting	Description

`Toolx = ToolName; Commandline Parameter`	Enter the name and command line to define the tools.
	Example:
	`Tool1 = CCLaunch; CCLaunch.exe`
	`Tool2 = SFCSAVE; d:\src\sfcsave\sfcsave.exe /I=d:\src\sfcsave\sfcsave.ini /P=%PRJDIR%PRJNAME.prj /M=%PLCADDR /F=%PRJDIR%PRJNAME.sfc`

NOTE: File names must be assigned in the 16 bit (8.3) file name format. This means that no more than 8 characters may be used for the file name and no more than 3 characters for the file extension or file type.

Q.2 Settings in the Projectname.INI File

Overview

This section describes the settings in the Projectname.INI file.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information for Projectname.INI File	1121
INI Settings for the Event Viewer [Online Events]	1122
INI-Settings for the Online-Backup [Backup]	1123

General Information for Projectname.INI File

Introduction

Project specific software settings can be made in the Projectname.INI file. The file is either created automatically by Concept (after configuration changes) or can be created by the user. Make sure that the file name always contains the respective project names, e.g. TESTPRJ.INI. The file can contain preset values which can then be changed.

Where is the Projectname.INI file situated?

The Projectname.INI file must be in the Concept project directory , e.g. C:\CONCEPT\TESTPRJ\TESTPRJ.INI

Editing the INI File

The different settings are separated in keywords, e.g. **[Configurator]**. To edit the INI file, enter the command line with the value or the path for the keyword (see *INI Settings for the Event Viewer [Online Events], page 1122*). Then save the created or modified file.

NOTE: Changes to the INI file are accepted directly by Concept.

INI Settings for the Event Viewer [Online Events]

Event Viewer [Online Events]

Specifying a user defined error description:

Setting	Description

`Error code="Error description"`
Example:
`-2676="Error in process D"`
	The defined error description is assigned to the error code.
	Note: The error code is entered in the event viewer (in the main menu **Online**).

`Parameter value="Error description"`
Example:
`62860="Error in process B"`
	The defined error description is assigned to the parameter value.
	Note: For EFB ONLEVT, error code -2696 is always used. Therefore the value at the PARAM input is selected for the assignment of the error description.

INI-Settings for the Online-Backup [Backup]

Define path and backup files

Example 1

Entry in the Projekt.INI-file:

```
Command=backup.bat  d:\src\prj1  \\sg-cc\prj\prj1
```

The entry contains the generated *.BAT- or *.EXE-file, and the source and target path of the backup files.

Entry in the batch-file backup.bat:

```
Rem Copy complete project with all subfolders
xcopy %1 %2 /s
```

All backup-files and sub-directories of the project are saved in the specified path. `%1` and `%2` refer to the source and target path in the entry of the Projekt.INI-file.

Note: For the path data of the target, the UNC-Notation can be used.

Example 2

Entry in the Projekt.INI-file:

```
Command=backup.bat
```

The entry contains the generated *.BAT- or *.EXE-file. The source and target path of the backup-files is located in the *.BAT- or *.EXE-file.

Entry in the batch-file backup.bat:

```
Rem Copy Project-, Var- and Diag files to remote PC
xcopy  d:\src\prj1\*.*           \\sg-cc\prj\prj1
xcopy  d:\src\prj1\Var\*.*       \\sg-cc\prj\prj1\Var
xcopy  d:\src\prj1\myprj.DIA\*.* \\sg-cc\prj\prj1\myprj.DIA
```

Backup files from the project directory and the sub-directories "Var" and "myprj.DIA" are saved in the paths specified.

Note: For the path data of the target, the UNC-Notation can be used.

Interrupt Processing

R

Overview

This chapter describes Interrupt processing handling with Quantum and Concept IEC.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
R.1	General information about interrupt sections	1126
R.2	Interrupt section: Timer event section	1128
R.3	Interrupt section: I/O event section	1141
R.4	Modules for interrupt sections	1146

R.1 General information about interrupt sections

General Information about Interrupt Processing

At a Glance

Starting from Concept version 2.6 together with the Quantum modules 140-CPU-434 A or 140-CPU-534 14 A/B and the 140-HLI-340-00 if required, Interrupt processing functions are made available for the configuration of IEC conforming programs. Special Interrupt sections allows the creation of both Time interrupts (Timer event sections) and I/O interrupts (I/O event sections).

The following Interrupt processing is possible:

- **Timer event sections**
  - Timer event sections enable program sections to be processed in constant, programmable time intervals. The internal time interrupt is used for this.
    To determine the time interrupts, every Timer event section is assigned a constant time value for their execution (scan rate) in a range from 10 ms to 1023s in the **Section Properties for Timer Event Sections** (in the **File** main menu) dialog box. Optimal runtime behavior can be configured using the Interval, Time base and Phase parameters (moving the execution to another cycle with the same interval). The processing of a cyclic section is immediately stopped if a time interrupt occurs. After the Timer event section is processed, the program execution is continued from the point where it was stopped.
    Using the direct I/O blocks IMIO_IN and IMIO_OUT on these sections enables current inputs and outputs to be processed at predetermined intervals. Time critical problems can be easily resolved, i.e. the realization of control loops in closed loop control systems.
    The maximum permitted execution duration for a Timer event section is 20 ms!

- **I/O Event sections**
  - I/O event sections enable program parts to be processed according to a signal change on a specific Hardware interrupt input. The Interrupts required for this (spontaneous I/O) are generated by the 140-HLI-340-00 module.
    Every I/O event section is assigned a pin (input) on the 140-HLI-340-00 module via the **Section Properties for I/O Event Sections** (in the **File** main menu) dialog box. Depending on the parameters set in the Configurator, the signal change on pin (0->1, 1->0 or 0->1 and 1->0) triggers a hardware interrupt on the CPU logic processor. The processing of a cyclic section or Timer event section is immediately stopped if a hardware interrupt occurs. After the I/O event section is processed, the program execution is continued from the point where it was stopped.
    Using the direct I/O blocks IMIO_IN and IMIO_OUT on these sections enables spontaneous outputs (and further inputs) to be processed extremely quickly. A critical event can be reacted to immediately, i.e. independently from the cycle.
    Special EFBs for Interrupt sections allow among other things, the program dependent disable/enable of Interrupt sections.
    The maximum permitted execution duration for an I/O event section is 20 ms!

NOTE: The direct I/O blocks IMIO_IN and IMIO_OUT only work when the corresponding I/O module is installed on the local backplane or backplane expander!

Limitations

Interrupt sections cannot be used together with the following functions:

- Hot Standby
  - If an additional Hot Standby is configured in a project, an error message is returned!
- ULEX/ASUP with modules 140-NOA-611-00, 140-NOA-611-10 and 140-ESI-062-00
  - The 140-NOA-622-00 can be used together with Interrupt sections instead of the 140-NOA-611-x0.
- LL984 sections

R.2 Interrupt section: Timer event section

Overview

This chapter contains a description for timer event sections.

What's in this Section?

This section contains the following topics:

Topic	Page
Timer Event Sections	1129
Defining the Scan Rate	1130
Defining the Phase	1131
Execution Order	1134
Operating System	1135
Examples for Parameterization	1137

Timer Event Sections

Introduction

Timer event sections are created in the same way as cyclic sections using the **File** →**New Section...** menu command. A maximum of 16 Timer event sections can be created. A Timer event section can only be selected when a CPU 140-CPU-434 or 140 CPU 534 is configured in the Configurator.

The CPU Hardware 140 CPU 434 12 A or 140 CPU 534 14 A/B is required to execute a program with Timer event sections!

A new **Timer Events** group is created automatically for Timer event sections where the Timer event sections are displayed. This group is placed in before the cyclic sections and after the **I/O Events** group in the project browser (see *Execution Order, page 1134*).

Timer event sections are programmed principally as with cyclic sections (see *Step 3: Creating the User Program, page 85*), only the selection of existing EFBs is limited.

Blocks (EFB) not available in Timer event sections:
- **F_TRIG**, **R_TRIG** (IEC library, group: Edge Detection)
- **TOF**, **TON**, **TP** (IEC library, group: Timer)
- **ERR2HMI**, **ERRMSG** (DIAGNO library, group: Diag View)
- **ACT_DIA**, **DYN_DIA**, **GRP_DIA**, **LOCK_DIA**, **PRE_DIA**, **REA_DIA** (DIAGNO library, group: Diagnostics)
- **XACT**, **XACT_DIA**, **XDYN_DIA**, **XGRP_DIA**, **XLOCK_DIA**, **XLOCK**, **XPRE_DIA**, **XREA_DIA**, (DIAGNO library, group: Extended)

Timer event section parameters are set in the **Section Properties for Timer Event Sections** dialog box using the **Interval**, **Time Base** and **Phase** parameters. This enables you to specify at what intervals (scan rate) the sections is processed.

Simultaneously created Timer event sections, for example with the same scan rate, are processed consecutively according to the execution order and priority in the same cycle. The runtimes of these Timer event sections are added in this cycle and make it longer. This can be avoided by using a time delayed execution of the section (phase) which allows a more constant total cycle time to be achieved.

After the program is started, the execution of the 1st Timer event section is delayed by 1 second!

Defining the Scan Rate

Description

Using the entries **Time Base** and **Interval**, it is possible to define nearly any scan rate for a timer event section.

Selectable time base:

- 10 ms
- 100 ms
- 1 s

Interval values:

- whole number multiples of the time base in the range from 1 to 1023

scan rate = interval * time base

- can be defined in range from 10 ms to 1023 s
- can be set in steps that correspond to the selected time base

NOTE: Changing section properties in online mode caused the CHANGED state. That means a **Download Changes** must take place so that the state is EQUAL again. When changing the phase, the PLC also has to be stopped and started again (**Online →Online Control Panel**) to accept the changes.

Examples

Example 1:

Required scan rate = 0.310 s (310 ms)

Scan rate (ms)	Interval	Time base (ms)
310	31	10

For a scan rate of 0.31 s, define a value of 31 for the interval

Example 2:

Required scan rate = 0.3 s (300 ms)

Scan rate (ms)	Interval	Time base (ms)
300	30	10
300	3	100

For a scan rate of 0.3 s, a value of 30 or 3 can be defined for the interval depending on the selected time base.

Basically, any resulting setting can be selected. However, the possible setting should take the phase into consideration (see *Defining the Phase, page 1131*).

Defining the Phase

Description

To prevent several timer event sections from being processed in the same cycle, they can be assigned different phase values.

Phase values:
- whole number multiples of the time base
- Range from 0 to interval -1, max. 1022

NOTE: Changing section properties in online mode caused the CHANGED state. That means a **Download Changes** must take place so that the state is EQUAL again. When changing the phase, the PLC also has to be stopped and started again (**Online →Online Control Panel**) to accept the changes.

Examples

Example 3:

Scan rate (ms)	Interval	Time base (ms)	Phases	Max. number of time displaced timer event sections
300	30	10	0...29	30
300	3	100	0...2	3

Example 4:

Several timer event sections with 300 ms scan rate (see Example 3)

Interval	Time base (ms)	Phase
30	10	0 (defined for all sections)

Result: All sections are processed in the same cycle, i.e. the program cycle time increased by the sum of the runtimes for all sections to be executed every 300 ms!

Example 5:

3 to 16 timer event sections with 300 ms scan rate (see Example 3)

1. Section

Interval	Time base (ms)	Phase
30	10	0

2. Section

Interval	Time base (ms)	Phase
30	10	1

3. Section

Interval	Time base (ms)	Phase
30	10	2

... Section

Interval	Time base (ms)	Phase
30	10	...

16. Section

Interval	Time base (ms)	Phase
30	10	15

Result: After the program is started, the 1st execution takes place for the
- 2nd timer event section (phase 1) after 1s+1*10 ms = **1s+10 ms**
- 3rd timer event section (phase 2) after 1s+2*10 ms = **1s+20 ms**
- ...
- 16. timer event section (phase 15) after 1s+15*10 ms = **1s+150 ms**
- 1st timer event section (phase 0) after 1s+30*10 ms = **1s+300 ms**

The second execution after the program start takes place for the
- 2nd timer event section (phase 1) after 1s+300 ms+1*10 ms = **1s+310 ms**
- 3rd timer event section (phase 2) after 1s+300 ms+2*10 ms = **1s+320 ms**
- ...
- 16th timer event section (phase 15) after 1s+300 ms+15*10 ms = **1s+450 ms**
- 1st timer event section (phase 0) after 1s+300 ms+30*10 ms = **1s+600 ms**

Each further execution of a timer event section takes place after exactly 300 ms, i.e. the runtimes of the (max. 16) timer event sections are distributed over (max. 30 selectable) different program cycles.

Explanation of example 5

If a time base of 10 ms is selected in example 5 (phase 0...29), the maximum number of 16 timer event sections can be executed using time displacement. That means a time displacement between 10 and 300 ms per section can be selected in steps of 10 ms. Each of the maximum possible 16 timer event sections is executed in a different program cycle. Every 10...20 ms, a program cycle extended by the execution duration of a timer event section occurs.

If a time base of 100 ms is selected in example 5 (phase 0, 1 and 2), the execution of the max. 16 possible timer event sections is distributed over only a max. of 3 time displaced program cycles. In these 3 program cycles, several timer event sections must be executed one after the other. Every 100 ms, a program cycle extended by the sum of the execution times of several timer event sections occurs!

The user should use the time base and phase to guarantee evenly distributed timer event sections (see timing diagram).)

Execution Order

Description

When creating the first timer event section, a new **Timer Events** group is created automatically in which the new section appears. This group is placed in before the cyclic sections and after the **I/O Events** group in the project browser. The next timer event section to be created is automatically always placed at the end in the group **Timer Events**.

Priority

Timer event sections do not have priorities set between them, i.e. they cannot be interrupted by another timer event section.

If several timer event sections are triggered at the same time in a program cycle, they are executed consecutively according to the order in which they were created.

However, an *I/O Event Sections, page 1142* has a higher priority and therefore interrupts a timer event section. The interrupted timer event section is only returned to after the execution of the I/O section is completed.

Operating System

Setting parameters according to runtime aspects

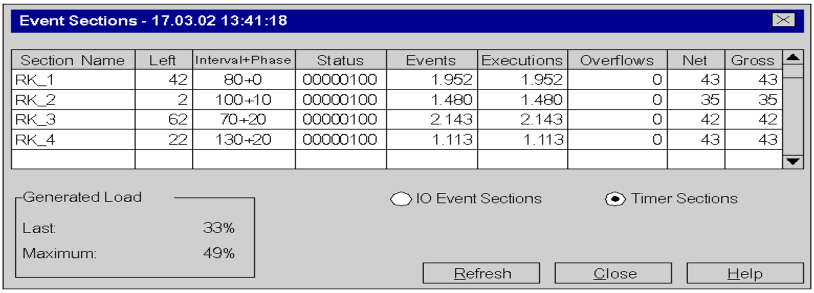
Take the following into consideration when setting parameters for timer event sections:

1. The runtime for a timer event section can be a maximum of 20 ms (see also *Runtime Error, page 1135*).
2. The scan rate (interval * time base) must be larger than the runtime for the timer event section.
3. Select the phase in a program cycle so that only one timer event section is executed whenever possible.
4. Take note of the distance between phases for time base 10 ms and a runtime for the timer event section > 10 ms! (Select a distance between phases >1 to prevent runtime overlaps.)
5. For optimal processor load:
   The execution of all timer event sections must be evenly distributed by selecting a suitable phase using the time for the scan rates.
6. Sufficient time must be remaining for the execution of the cyclic sections so that the cyclic I/O is handled in acceptable intervals!
7. Execution of inputs and outputs as direct I/O modules IMIO_IN and IMIO_OUT in the timer event section. For example if the uneven intervals for the cyclic I/O are not sufficient to create a control loop.
8. Create a timing diagram (see *Examples for Parameterization, page 1137*):
   This makes it possible to determine the optimal phase, as well as the actual time intervals for the cyclic I/O.
9. Do not create all control loops as timer event sections:
   Control loops can also be programmed as cyclic sections using the necessary high-speed CPUs and the SAMPLETM module!

Runtime Error

Any errors that occur when processing the program, e.g. runtime is exceeded, overflow, etc. are shown in a table in the **Event Sections** dialog box (in the **Online** main menu).

The following table is based on *Example 4: Control loops with different scan rates, with phases, constant cycle time, page 1140* in section "Examples for Setting Parameters":



Event Sections - 17.03.02 13:41:18

Section Name	Left	Interval+Phase	Status	Events	Executions	Overflows	Net	Gross
RK_1	42	80+0	00000100	1.952	1.952	0	43	43
RK_2	2	100+10	00000100	1.480	1.480	0	35	35
RK_3	62	70+20	00000100	2.143	2.143	0	42	42
RK_4	22	130+20	00000100	1.113	1.113	0	43	43

Generated Load

Last: 33%

Maximum: 49%

○ IO Event Sections

● Timer Sections

Runtime exceeded for a timer event section

If the runtime for a timer event section is >20 ms, the following process is carried out:

1. In the **Event Sections** table, status bit 2 is set (watchdog timer has expired)

2. The timer event section is disabled.

Carry out the following steps to detect when the runtime is exceeded in timer event sections:

Step	Action
1	Using the ISECT_STAT function block.
2	Activate the mode to display enable states in the project browser. Then the symbols for the disabled sections are marked red.
3	Call the status table in **Online** →**Event Sections**.

Carry out the following steps if an timer event section is disabled:

Step	Action
1	Reduce the runtime of the timer event section to <20 ms.
2	Enabling a timer event section Examples: ● In the project browser, activate the **Switch enable state** command. ● Programming: 0 -> sectname.disable **Caution:** If the runtime error still occurs, the timer event section is not processed even though the section symbol is marked green in the project browser!
3	After enabling the timer event section, the RESET function block parameter on the ISECT_STAT EFB must be set. Only then are current values shown in the status table (**Online** →**Event Sections**).

Examples for Parameterization

Introduction

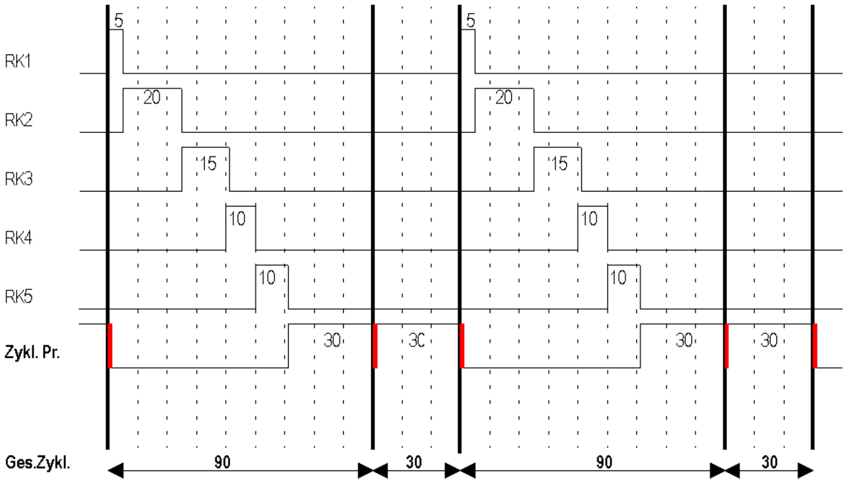
The examples shown here with the values given represent theoretical information and should mainly be used to clarify the effects of various phase values and distances between phases on the entire system. With (preset) values determined using timing diagrams and tests, the user can establish a predictive view and reach an optimal distribution of the timer event sections and prevent runtime overflows.

Example 1: Control loops with the same scan rates, all phases = 0

Preset values:

Preset values	**Scan rate**	**Time base**	**Interval**	**Runtime**	**Phase**
Cyclic Program				30	
RK 1	120 ms	10 ms	12	5	0
RK 2	120 ms	10 ms	12	<20	0
RK 3	120 ms	10 ms	12	15	0
RK 4	120 ms	10 ms	12	10	0
RK 5	120 ms	10 ms	12	10	0

Timing diagram (times in ms)



RK Control loop

Cyclic Program Cyclic Program
Total Cycle Time Total Cycle Time

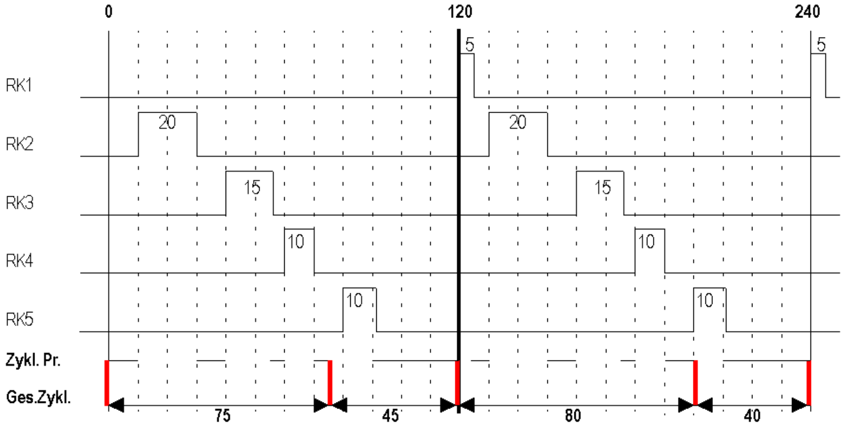
NOTE: The total cycle time switches between 90 and 30 ms.

Example 2: Control loops with the same scan rates, with phases, minimum distance between phases

Preset values:

Preset values	Scan rate	Time base	Interval	Runtime	Minimum distance between phases	Phase
Cyclic Program				30		
RK 1	120 ms	10 ms	12	5		0
RK 2	120 ms	10 ms	12	<20	5<10 ms->+1	1
RK 3	120 ms	10 ms	12	15	20<30 ms->+3	4
RK 4	120 ms	10 ms	12	10	15<20 ms->+2	6
RK 5	120 ms	10 ms	12	10	10<20 ms->+2	8

Timing diagram (times in ms)



RK Control loop
Cyclic Program Cyclic Program
Total Cycle Time Total Cycle Time

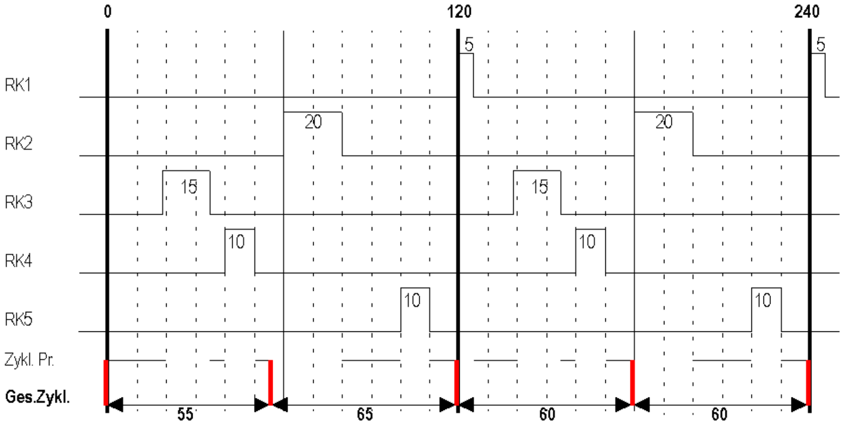
NOTE: The total cycle time (except for the first cycle) switches between 80 and 40 ms.

Example 3: Control loops with the same scan rates, with phases, constant cycle time

Preset values:

Preset values	Scan rate	Time base	Interval	Runtime	Phase
Cyclic Program				30	
RK 1	120 ms	10 ms	12	5	0
RK 2	120 ms	10 ms	12	<20	6
RK 3	120 ms	10 ms	12	15	2
RK 4	120 ms	10 ms	12	10	4
RK 5	120 ms	10 ms	12	10	10

Timing diagram (times in ms)



RK Control loop
Cyclic Program Cyclic Program
Total Cycle Time Total Cycle Time

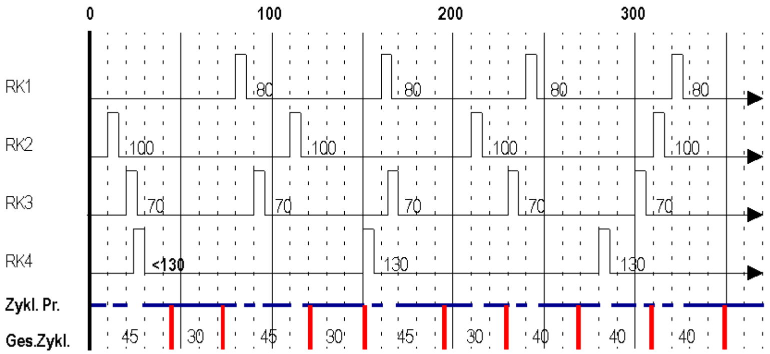
NOTE: The total cycle time (except for the first cycle) is always 60 ms.

Example 4: Control loops with different scan rates, with phases, constant cycle time

Preset values:

Preset values	Scan rate	Time base	Interval	Runtime	Phase
Cyclic Program				30	
RK 1	80 ms	10 ms	8	5	0
RK 2	100 ms	10 ms	10	5	1
RK 3	70 ms	10 ms	7	5	2
RK 4	130 ms	10 ms	13	5	2

Timing diagram (times in ms)



RK Control loop
Cyclic Program Cyclic Program
Total Cycle Time Total Cycle Time

NOTE: The timing diagram beginning here shows a favorable distribution of the execution of all sections. The cyclic I/O is also handled in predictable intervals. However, overlapping of individual runtimes cannot be ruled out.

R.3 Interrupt section: I/O event section

Overview

This chapter contains a description for I/O event sections.

What's in this Section?

This section contains the following topics:

Topic	Page
I/O Event Sections	1142
Priority	1143
Runtime Error	1144

I/O Event Sections

Introduction

An I/O event section is carried out depending on the hardware interrupts of a 140-HLI-340-00.

The 140-HLI-340-00 module is equipped with 16 inputs that can be configured as either fast inputs or interrupt inputs. Only interrupt inputs can trigger the execution of an I/O event section with the edge defined. Parameters must be set for the 140-HLI-340-00 module in the PLC Configuration accordingly.

I/O event sections are created in the same way as cyclic sections using the **File → New Section...** menu command. An I/O event section can only be selected when a CPU 140-CPU-434 oder 140-CPU-534 is configured in the Configurator. The CPU Hardware 140 CPU 434 12 A or 140 CPU 534 14 A/B is required to execute a program with I/O event sections!

A maximum of 64 I/O event sections can be created. The required hardware interrupts can be created by more than 4 HLI modules.

When creating the first I/O event section, a new **I/O Events** group is created automatically in which the new section appears. The **I/O Events** group appears first in the project browser, i.e. before the **Timer Events** groups and the cyclic sections. Every additional I/O event section to be created is automatically placed in the **I/O Events** group according to their priority (from top to bottom).

An execution order has no relevance for I/O event sections, since these sections can only be processed when a hardware interrupt occurs.

An I/O event section can only be interrupted by hardware interrupts with higher *Priority, page 1143*.

I/O event sections are programmed principally as with cyclic sections (see *Step 3: Creating the User Program, page 85*), only the selection of existing EFBs is limited.

Blocks (EFB) not available in Timer event sections:

- **F_TRIG**, **R_TRIG** (IEC library, group: Edge Detection)
- **TOF**, **TON**, **TP** (IEC library, group: Timer)
- **ERR2HMI**, **ERRMSG** (DIAGNO library, group: Diag View)
- **ACT_DIA**, **DYN_DIA**, **GRP_DIA**, **LOCK_DIA**, **PRE_DIA**, **REA_DIA** (DIAGNO library, group: Diagnostics)
- **XACT**, **XACT_DIA**, **XDYN_DIA**, **XGRP_DIA**, **XLOCK_DIA**, **XLOCK**, **XPRE_DIA**, **XREA_DIA**, (DIAGNO library, group: Extended)

I/O event section parameters are set in the **Section Properties for I/O Event Sections** dialog box using the **Slot** and **Input Pin** parameters. The **Slot** entry defines the slot on the local backplane when the 140-HLI-340-00 module is positioned for the triggered interrupt. The **Input Pin** defines pin number (1 to 16) for the 140-HLI-340-00 inputs that trigger the section processing.

Priority

Description

I/O event sections have priorities set between them. An active I/O event section can be interrupted by an I/O event section with higher priority. The interrupted section is continued after the section with higher priority has been processed.

If other interrupts with lower priority occur while processing an I/O event section, the active I/O event section is not interrupted. However, these interrupt signals are saved and the respective sections are processed according to their priority when the active I/O event section is complete. If an interrupt which is saved but not yet processed occurs again, the second interrupt is lost . The overflow counter is incremented (see the table in the **Event Sections** dialog box in the **Online** main menu).

The priority of an I/O event section is determined by the position of an input pin on the 140-HLI-340-00 module in the local backplane. Therefore:

The lower the slot address and the lower the pin number, the higher the priority. The slot and the input pin number of an I/O event section is assigned in the **Section Properties for I/O Event Sections** dialog box.

Example 1:

Priority	Slot	Input Pin
Higher	1	5
Lower	6	1

Example 2:

Priority	Slot	Input Pin
Higher	3	5
Lower	3	6

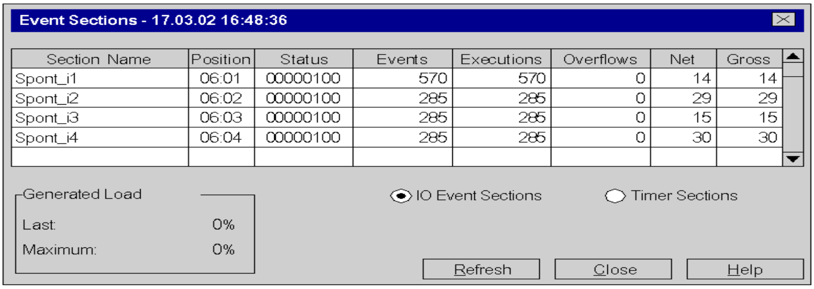
NOTE: I/O event sections can be interrupted over several priority levels (interrupt in interrupt), therefore the total cycle can be greatly increased.

Runtime Error

Description

Any errors that occur when processing the program, e.g. runtime is exceeded, overflow, etc. are shown in a table in the **Event Sections** dialog box (in the **Online** main menu).

Table in the dialog box **Event Sections**:



Event Sections - 17.03.02 16:48:36

Section Name	Position	Status	Events	Executions	Overflows	Net	Gross
Spont_i1	06:01	00000100	570	570	0	14	14
Spont_i2	06:02	00000100	285	285	0	29	29
Spont_i3	06:03	00000100	285	285	0	15	15
Spont_i4	06:04	00000100	285	285	0	30	30

Generated Load

Last: 0%

Maximum: 0%

● IO Event Sections ○ Timer Sections

Runtime exceeded for an I/O event section

If the runtime for a timer event section is >20 ms, the following process is carried out:

1. In the **Event Sections** table, status bit 2 is set (watchdog timer has expired)
2. The I/O event section is disabled.

Carry out the following steps to detect when the runtime is exceeded in I/O event sections:

Step	Action
1	Using the ISECT_STAT function block.
2	Activate the mode to display enable states in the project browser. Then the symbols for the disabled sections are marked red.
3	Call the status table in **Online** →**Event Sections**.

Carry out the following steps if an I/O event section is disabled:

Step	Action
1	Reduce the runtime of the I/O event section to <20 ms.
2	Enable the I/O event section. Examples: ● In the project browser, activate the **Switch enable state** command. ● Programming: 0 -> sectname.disable **Caution:** If the runtime error still occurs, the I/O event section is not processed even though the section symbol is marked green in the project browser!
3	After enabling the I/O event section, the RESET function block parameter on the ISECT_STAT EFB must be set. Only then are current values shown in the status table (**Online** →**Event Sections**).

R.4 Modules for interrupt sections

EFBs for Interrupt Sections

EFBs to disable and enable interrupt sections

The following function blocks are available:

- ISECT_OFF
  - The ISECT_OFF block can be used to disable a specific I/O event section or Timer event section, i.e. the interrupt has no effect on this Interrupt section. The name of the section to be disabled is defined by the SECT_CTRL data type variable entered at the input. This variable is automatically created for each section.
- ISECT_ON
  - The ISECT_ON block can be used to enable a specific I/O event section or Timer event section, i.e. the interrupt has effect on this Interrupt section again. The name of the section to be enabled is defined by the SECT_CTRL data type variable entered at the input. This variable is automatically created for each section.
- I_LOCK
  - The I_LOCK block is used to disable all I/O event sections or Timer event sections, i.e. the interrupts have no effect on Interrupt sections.
- I_UNLOCK
  - The I_UNLOCK block is used to enable all I/O event sections or Timer event sections, i.e. the interrupts have effect on the respective Interrupt sections.

Other EFBs for Interrupt sections

The following function blocks are available:

- ISECT_STAT
  - With the ISECT_STAT block, the status (see **Event Sections**dialog box) of a section can be read and evaluated by the program.
- I_MOVE
  - The I_MOVE block prevents an interruption of value assignments from an input to an output by an interrupt. This means the processing of an I_MOVE is not interrupted by an interrupt. This enables data consistency between an input and an output if the variable is used both in cyclic as well as in interrupt sections. The MOVE block has the same function, but value assignment is not interrupt protected (for further details see the block description).

Automatic Connection to the PLC

S

Overview

This chapter is a description of both methods of automatically connecting with a PLC.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Automatic Connection with Command Line Parameters (Modbus, Modbus +, TCP/IP)	1148
Automatic Connection with the CCLaunch Tool (Modbus Plus)	1151

Automatic Connection with Command Line Parameters (Modbus, Modbus +, TCP/IP)

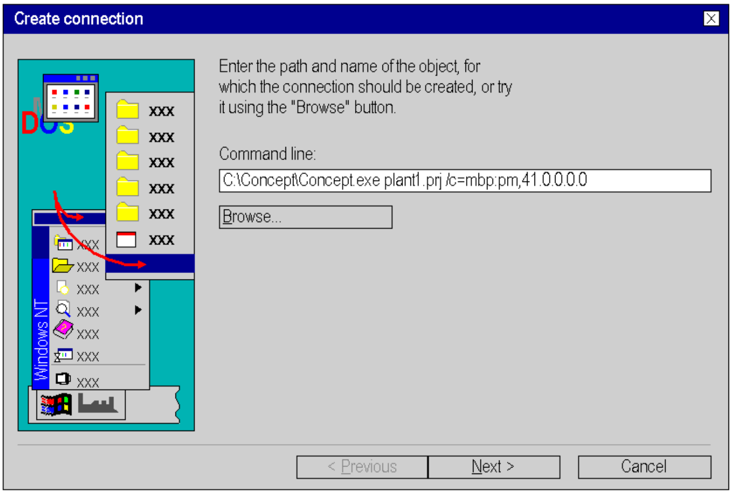
At a Glance

You can make a connection with any PLC automatically by using the command line of the Windows dialog **Create Shortcut** and entering the Modbus Plus Routing path.

NOTE: If, in dialog **Common Preferences** option **Connect to controller at Startup** (**Options** →**Preferences** →**Common...**) is activated then the extended parameters for the connection to the PLC are priority.

The command line parameters for automatically connecting are added to the end of the command line parameters for the project symbol (see following image).

Dialog box: **Create connection**



Create connection

Enter the path and name of the object, for which the connection should be created, or try it using the "Browse" button.

Command line:

C:\Concept\Concept.exe plant1.prj /c=mbp:pm,41.0.0.0.0

Creating an Automatic Connection with Command Line Parameters

The procedure for connecting automatically using command line parameters is as follows:

Step	Action
1	Go to **Start** →**Settings** →**Taskbar...** and open the dialog **Taskbar Properties**.
2	In the register **Start Menu Programs**/**Expanded** (Win2000), select the **Add...** command button.
3	In the **Create Shortcut** dialog box, select the **Browse...** command button.
4	In the **Browse** dialog box, go to the Concept installation path and double-click on the file **CONCEPT.EXE**. **Reaction:** The **Browse** dialog box is closed and the file **CONCEPT.EXE** including the path is entered in the **command line:** E.g. **C:\CONCEPT\CONCEPT.EXE**.
5	Now, add the project name of the project in the command line, e.g. **C:\CONCEPT\CONCEPT.EXE PLANT1.PRJ**.
6	Now add the Modbus Plus-Routing path of the PLC to the command line, e.g. **C:\CONCEPT\CONCEPT.EXE PLANT1.PRJ /c=mbp : 41.0.0.0.0** and confirm your entries with the **Next >**button. **Note:** A definition of the various command line parameters can be found in section *Definition of Command Line Parameters, page 1150*.
7	In the **Select program group** dialog box, select an existing program group for the symbol or create a new one using **New folder...** . Confirm the entry using the **Next >**command button.
8	In the **Select program designation** dialog box, select the project name and confirm using the **Finish** command button.
9	Close the **Taskbar Properties** dialog box with **OK**. **Reaction:** The properties dialog box is closed and the project symbol is available in the start menu of the folder you selected.
10	Open the folder with the project symbol in the Start menu. Select the project symbol and click the right mouse button. **Reaction:** A menu window is opened.
11	Select the **Properties** command button. **Reaction:** The **"Project Symbol Name" Properties** dialog box is opened.
12	Go to the **Connection** register and complete the command line **Working directory**/**Target** (Win2000) with the name of the project directory, e.g. **C:\CONCEPT\PROJECTS**. Confirm the entry using the **Apply** command button.
13	Then exit the dialog box by selecting **OK**.
14	Open the project by clicking on the project symbol. **Reaction:** Concept reads the defined Modbus Plus Routing path and automatically creates a connection to the PLC.

Definition of Command Line Parameters

The command line parameters contain the PLC address and the protocol type (Modbus, Modbus Plus, TCP/IP).

Command line parameters for Modbus:

Protocol type	**Command line parameters**	**Meaning**
Modbus	`/c=[x,]mb:p[, m]`	**x** - Serial connection (COM): 1 - 4, optional, default is 1 **p** - PLC address: 0 - 255 **m** - Modus for device communication: RTU/ASCII, optional, default is RTU **Note:** The settings (Baud Rate, Data Bits, Parity, Stop Bits) for the serial connection (COM) are made in the **Connect to PLC** dialog box.

Example:

```
c:\concept\concept.exe  plant1.prj  /c=2,mb:001,ASCII
```

Command line parameters for Modbus Plus:

Protocol type	**Command line parameters**	**Meaning**
Modbus Plus	`/c=[x,]mbp: n.n.n.n.n`	**x** - Modbus Plus connection: 0 -1, optional, default is 0 **n** - Routing path (PLC address): 0 - 64

Example:

```
c:\concept\concept.exe  plant1.prj  /c=mbp:41.0.0.0.0
```

Command line parameters for TCP/IP:

Protocol type	**Command line parameters**	**Meaning**

TCP/IP	`/c=[x,]mbt:m.m.m.m`
	`/c=[x,]mbt:HostName`
		x - Bridge Modbus Plus Index: 0 -255, optional, default is 0
		m - IP address: 0 - 255
		HostName - e.g. for the PLCSIM32, the HostName = Localhost

Example:

```
c:\concept\concept.exe  plant1.prj  /c=mbt:139.158.107.9
c:\concept\concept.exe  plant.prj  /c=mbt:Localhost
```

Disadvantage

In a large Modbus Plus Network, a separate command line is required for each PLC. If an address changes at any time (e.g. Bridge address the command line parameters must be redone for every programming device that makes access.

Automatic Connection with the CCLaunch Tool (Modbus Plus)

At a Glance

You can use the CCLaunch tool to create a complete Routing path, which then creates a connection to the PLC in the corresponding Modbus Plus segment automatically.

The CCLaunch tool can also only be used to open the assigned project for making changes.

The CCLaunch tool is executed with the CCLAUNCH.EXE file in the Concept directory.

Selection condition

The CCLaunch tool can only be used if a topology file exists and the path is entered in the CCLaunch tool.

Creating an Automatic Connection

Create an ASCII file (topology file) for the automatic connection to the PLC and name it e.g. CCLEXAMP.TXT. Define all Routing paths and segment names of the entire project network in this *.TXT file. Then, copy the file to your server so that every programming device can access it. Compared with a command line parameter entry *(see page 1148)*, you have the advantage of only having to change one file when an address (e.g. bridge address) changes in the routing path.

To activate the automatic connection, start the CCLaunch tool. Enter the path of the topology file, the path for the projects and the address of the Modbus Plus adapter one time only. The definitions will then remain until they are changed again by the user. Enter the start segment that must be defined for the programming device (PC). Define the target segment that must be defined for the PLC to which it will be connected as another setting. Then select the PLC that you have defined in the topology file.

NOTE: For creating the automatic connection, the check box **Autoconnect to PLC** (**Options** →**Preferences** →**Commun...**) must be activated.

These entries allow CCLaunch to create a complete Routing path, which then creates a connection to the PLC automatically.

Opening Associated Project

To open the assigned project directly, define the same settings as for the automatic connection. Then activate the check box **Open Associated Project**.

NOTE: If you only want to open this project, deactivate the check box **Autoconnect to PLC**.

Creating the Topology File (*.TXT)

The topology file (*.TXT) only has to be created one time and it contains the description of the entire Modbus Plus network as well as an option description of the projects assigned to the PLC. These can then be stored centrally on the network/server.

The topology file (*.TXT) contains the two keywords [Segment] and [Routing]. The definition of the individual segment begins with keyword [Segment]. The definition of each individual Routing path begins with the keyword [Routing].

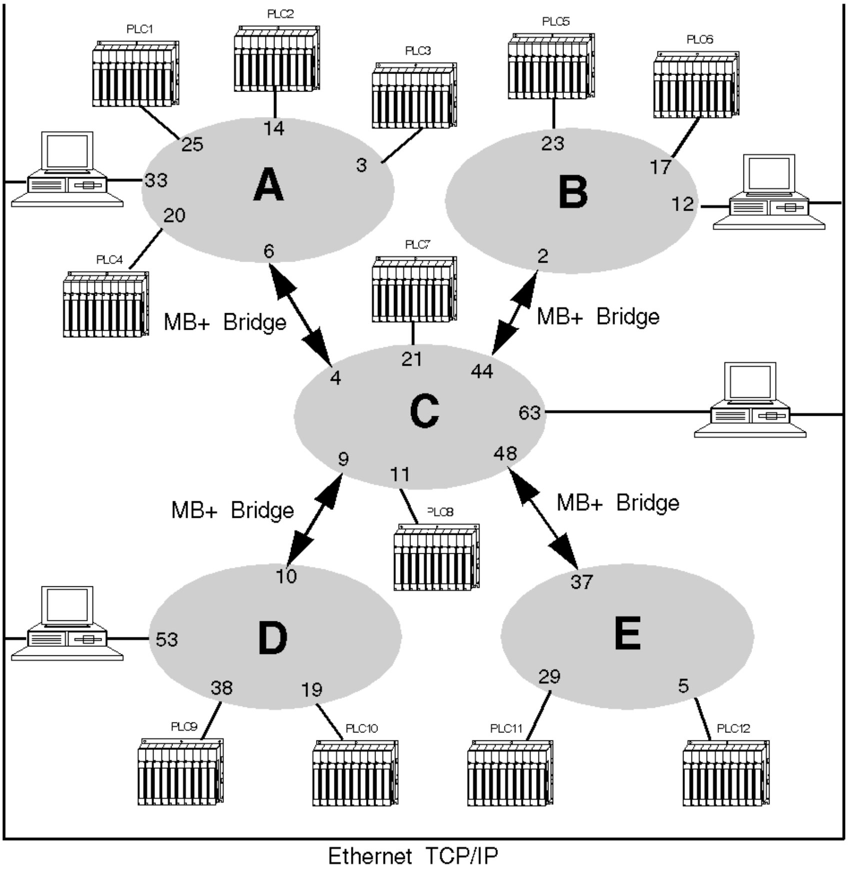
NOTE: For the definition of the PLC, the PLC name must be unique throughout the entire Modbus Plus network.

Example:

```
[Segment]="Segment name"

"PLC Name"="MB+Address" : "Project name" (optional)

"PLC Name"="MB+Address" : "Project name" (optional)

[Routing]SegmentX="Routing path"

[Routing]SegmentY="Routing path"
```

Example of a Topology File (*.TXT)

Example of a Modbus Plus network with different segments:



A Segment A
B Segment B
C Segment C
D Segment D
E Segment E

Contents of the Topology File (*.TXT)

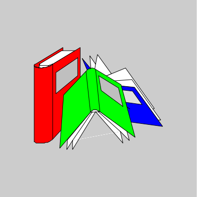
```
[Segment]=SegmentA
PLC1 = 25 : Project 1
PLC2 = 14 : Project 2
PLC3 = 3  : Project 3
PLC4 = 20 : Project 4
[Routing]  SegmentB=6.44
[Routing]  SegmentC=6
[Routing]  SegmentD=6.9
[Routing]  SegmentE=6.48
[Segment]=SegmentB
PLC5 = 23 : Project 5
PLC6 = 17 : Project 6
[Routing]  SegmentA=2.4
[Routing]  SegmentC=2
[Routing]  SegmentD=2.9
[Routing]  SegmentE=2.48
[Segment]=SegmentC
PLC7 = 21 : Project 7
PLC8 = 11 : Project 8
[Routing]  SegmentA=4
[Routing]  SegmentB=44
[Routing]  SegmentD=9
[Routing]  SegmentE=48
[Segment]=SegmentD
PLC9 = 38 : Project 9
PLC10 = 19: Project 10
[Routing]  SegmentA=10.4
[Routing]  SegmentB=10.44
[Routing]  SegmentC=10
[Routing]  SegmentE=10.48

[Segment]=SegmentE
PLC11 = 21: Project 11
PLC12 = 11: Project 12
[Routing]  SegmentA=37.4
[Routing]  SegmentB=37.44
[Routing]  SegmentC=37
[Routing]  SegmentD=37.9
```

Editing with the CCLaunch Tool

After creating the topology file (*.TXT), execute the following steps in the CCLaunch tool for the automatic connection:

Step	Action
1	Double click on the CCLAUNCH.EXE file in the Concept directory. **Reaction:** The CCLaunch tool is started.
2	Go to the **Settings** tab and define the path for the topology file (*.TXT) and the project path. **Note:** This is normally only defined once since this path should not have to be changed. This means that these settings only have to be made once and they remain saved until you change them for whatever reason. Example: **Topology file:** C:\CONCEPT\CONNECT\CCLEXAMP.TXT **Path for projects:** C:\CONCEPT\TESTPRJ\
3	Select the hardware address for the network connection in the **Modbus+ Port** field. **Note:** Whether this is port 0 or port 1 can be determined from the Windows system settings.
4	Select tab **Select PLC**, and enter the start segment, the target segment and the PLC that you want to connect with for the Routing path. Example: **Start Segment:** SegmentB **Dest. Segment:** SegmentE **PLC:** PLC8 In this example, the programming device is in segment B and should create a connection to the PLC with the name "PLC8" in segment E.
5	Go to the **Start Options** area and activate the check box **Autoconnect to PLC**.
6	Press the **Start Concept** button. **Reaction:** Concept reads the created Routing path and automatically creates a connection to the PLC.

Glossary



A

active Window

The window, which is currently selected. Only one window can be active at any given time. When a window is active, the color of the title bar changes, so that it is distinguishable from the other windows. Unselected windows are inactive.

Actual Parameters

Current connected Input / Output Parameters.

Addresses

(Direct) addresses are memory ranges on the PLC. They are located in the State RAM and can be assigned Input/Output modules.

The display/entry of direct addresses is possible in the following formats:

- Standard Format (400001)
- Separator Format (4:00001)
- Compact format (4:1)
- IEC Format (QW1)

ANL_IN

ANL_IN stands for the "Analog Input" data type and is used when processing analog values. The 3x-References for the configured analog input module, which were specified in the I/O component list, are automatically assigned to the data type and should therefore only be occupied with Unlocated Variables.

ANL_OUT

ANL_OUT stands for the "Analog Output" data type and is used when processing analog values. The 4x-References for the configured analog output module, which were specified in the I/O component list, are automatically assigned to the data type and should therefore only be occupied with Unlocated Variables.

ANY

In the present version, "ANY" covers the BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD elementary data types and related Derived Data Types.

ANY_BIT

In the present version, "ANY_BIT" covers the BOOL, BYTE and WORD data types.

ANY_ELEM

In the present version, "ANY_ELEM" covers the BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD data types.

ANY_INT

In the present version, "ANY_INT" covers the DINT, INT, UDINT and UINT data types.

ANY_NUM

In the present version, "ANY_NUM" covers the DINT, INT, REAL, UDINT and UINT data types.

ANY_REAL

In the present version, "ANY_REAL" covers the REAL data type.

Application Window

The window contains the workspace, menu bar and the tool bar for the application program. The name of the application program appears in the title bar. An application window can contain several Document windows. In Concept the application window corresponds to a Project.

Argument

Synonymous with Actual parameters.

ASCII-Mode

The ASCII (American Standard Code for Information Interchange) mode is used to communicate with various host devices. ASCII works with 7 data bits.

Atrium

The PC based Controller is located on a standard AT board, and can be operated within a host computer in an ISA bus slot. The module has a motherboard (requires SA85 driver) with two slots for PC104 daughter-boards. In this way, one PC104 daughter-board is used as a CPU and the other as the INTERBUS controller.

B

Backup file (Concept-EFB)

The backup file is a copy of the last Source coding file. The name of this backup file is "backup??.c" (this is assuming that you never have more than 100 copies of the source coding file). The first backup file has the name "backup00.c". If you have made alterations to the Definitions file which do not cause any changes to the EFB interface, the generation of a backup file can be stopped by editing the source coding file (**Objects** →**Source**). If a backup file is created, the source file can be entered as the name.

Base 16 literals

Base 16 literals are used to input whole number values into the hexadecimal system. The base must be denoted using the prefix 16#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

Example

16#F_F or 16#FF (decimal 255)

16#E_0 or 16#E0 (decimal 224)

Base 2 literals

Base 2 literals are used to input whole number values into the dual system. The base must be denoted using the prefix 2#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

Example

2#1111_1111 or 2#11111111 (decimal 255)

2#1110_0000 or 2#11100000 (decimal 224)

Base 8 literals

Base 8 literals are used to input whole number values in the octosystem. The base must be denoted using the prefix 8#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

Example

8#3_77 or 8#377 (decimal 255)

8#34_0 or 8#340 (decimal 224)

Binary Connections

Connections between FFB outputs and inputs with the data type BOOL.

Bit sequence

A data element, which consists of one or more bits.

BOOL

BOOL stands for the data type "boolean". The length of the data element is 1 bit (occupies 1 byte in the memory). The value range for the variables of this data type is 0 (FALSE) and 1 (TRUE).

Bridge

A bridge is a device which connects networks. It enables communication between nodes on two networks. Each network has its own token rotation sequence - the token is not transmitted via the bridge.

BYTE

BYTE stands for the data type "bit sequence 8". Entries are made as base 2 literal, base 8 literal or base 16 literal. The length of the data element is 8 bits. A numerical value range can not be assigned to this data type.

C

Clipboard

The clipboard is a temporary memory for cut or copied objects. These objects can be entered in sections. The contents of the clipboard are overwritten with each new cut or copy.

Coil

A coil is a LD element which transfers the status of the horizontal connection on its left side, unchanged, to the horizontal connection on its right side. In doing this, the status is saved in the relevant variable/direct address.

Compact format (4:1)

The first digit (the Reference) is separated from the address that follows by a colon (:) where the leading zeros are not specified.

Constants

Constants are Unlocated variables, which are allocated a value that cannot be modified by the logic program (write protected).

Contact

A contact is a LD element, which transfers a status on the horizontal link to its right side. This status comes from the boolean AND link of the status of the horizontal link on the left side, with the status of the relevant variable/direct address. A contact does not change the value of the relevant variable/direct address.

D

Data transfer settings

Settings which determine how information is transferred from your programming device to the PLC.

Data Types

The overview shows the data type hierarchy, as used for inputs and outputs of functions and function blocks. Generic data types are denoted using the prefix "ANY".

- ANY_ELEM
  - ANY_NUM
    ANY_REAL (REAL)
    ANY_INT (DINT, INT, UDINT, UINT)
  - ANY_BIT (BOOL, BYTE, WORD)
  - TIME
- System Data types (IEC Extensions)
- Derived (from "ANY" data types)

DCP I/O drop

A remote network with a super-ordinate PLC can be controlled using a Distributed Control Processor (D908). When using a D908 with remote PLC, the super-ordinate PLC considers the remote PLC as a remote I/O drop. The D908 and the remote PLC communicate via the system bus, whereby a high performance is achieved with minimum effect on the cycle time. The data exchange between the D908 and the super-ordinate PLC takes place via the remote I/O bus at 1.5Mb per second. A super-ordinate PLC can support up to 31 D908 processors (addresses 2-32).

DDE (Dynamic Data Exchange)

The DDE interface enables a dynamic data exchange between two programs in Windows. The user can also use the DDE interface in the extended monitor to call up their own display applications. With this interface, the user (i.e. the DDE client) can not only read data from the extended monitor (DDE server), but also write data to the PLC via the server. The user can therefore alter data directly in the PLC, while monitoring and analyzing results. When using this interface, the user can create their own "Graphic Tool", "Face Plate" or "Tuning Tool" and integrate it into the system. The tools can be written in any language, i.e. Visual Basic, Visual C++, which supports DDE. The tools are invoked when the user presses one of the buttons in the Extended Monitor dialog field. Concept Graphic Tool: Configuration signals can be displayed as a timing diagram using the DDE connection between Concept and Concept Graphic Tool.

Declaration

Mechanism for specifying the definition of a language element. A declaration usually covers the connection of an identifier to a language element and the assignment of attributes such as data types and algorithms.

Definitions file (Concept-EFB)

The definitions file contains general descriptive information on the selected EFB and its formal parameters.

Defragmenting

With defragmenting, unanticipated gaps (e.g. resulting from deleting unused variables) are removed from memory.

See also **PLC Selection** in the context help.

Derived Data Type

Derived data types are data types, which are derived from Elementary Data Types and/or other derived data types. The definition of the derived data types is found in the Concept data type editor.

A distinction is made between global data types and local data types.

Derived Function Block (DFB)

A derived function block represents the invocation of a derived function block type. Details of the graphic form of the invocation can be found in the "Functional block (instance)". In contrast to the invocation of EFB types, invocations of DFB types are denoted by double vertical lines on the left and right hand side of the rectangular block symbol.

The output side of a derived function block is created in FBD language, LD language, ST language, IL language, but only in the current version of the programming system. Derived functions can also not be defined in the current version.

A distinction is made between local and global DFBs.

DFB Code

The DFB code is the section's DFB code which can be executed. The size of the DFB code is mainly dependent upon the number of blocks in the section.

DFB instance data

The DFB instance data is internal data from the derived function blocks used in the program.

DINT

DINT stands for the data type "double length whole number (double integer)". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 32 bits. The value range for variables of this data type reaches from -2 exp (31) to 2 exp (31) -1.

Direct Representation

A method of displaying variables in the PLC program, from which the assignment to the logical memory can be directly - and indirectly to the physical memory - derived.

Document Window

A window within an application window. Several document windows can be open at the same time in an application window. However, only one document window can ever be active. Document windows in Concept are, for example, sections, the message window, the reference data editor and the PLC configuration.

DP (PROFIBUS)

DP = Remote Peripheral

Dummy

An empty file, which consists of a text heading with general file information, such as author, date of creation, EFB designation etc. The user must complete this dummy file with further entries.

DX Zoom

This property enables the user to connect to a programming object, to monitor and, if necessary change, its data value.

E

EFB code

The EFB code is the executable code of all EFBs used. In addition the used EFBs count in DFBs.

Elementary functions/function blocks (EFB)

Identifier for Functions or Function blocks, whose type definitions are not formulated in one of the IEC languages, i.e. whose body for example can not be modified with the DFB editor (Concept-DFB). EFB types are programmed in "C" and are prepared in a pre-compiled form using libraries.

EN / ENO (Enable / Error signal)

If the value of EN is equal to "0" when the FFB is invoked, the algorithms that are defined by the FFB will not be executed and all outputs keep their previous values. The value of ENO is in this case automatically set to "0". If the value of EN is equal to "1", when the FFB is invoked, the algorithms which are defined by the FFD will be executed. After the error-free execution of these algorithms, the value of ENO is automatically set to "1". If an error occurs during the execution of these algorithms, ENO is automatically set to "0". The output behavior of the FFB is independent of whether the FFBs are invoked without EN/ENO or with EN=1. If the EN/ENO display is switched on, it is imperative that the EN input is switched on. Otherwise, the FFB is not executed. The configuration of EN and ENO is switched on or off in the Block Properties dialog box. The dialog box can be invoked with the **Objects** → **Properties...**menu command or by double-clicking on the FFB.

Error

If an error is recognized during the processing of a FFB or a step (e.g. unauthorized input values or a time error), an error message appears, which can be seen using the **Online** →**Event Viewer...**menu command. For FFBs, the ENO output is now set to "0".

Evaluation

The process, through which a value is transmitted for a Function or for the output of a Function block during Program execution.

Expression

Expressions consist of operators and operands.

F

FFB (Functions/Function blocks)

Collective term for EFB (elementary functions/function blocks) and DFB (Derived function blocks)

Field variables

A variable, which is allocated a defined derived data type with the key word ARRAY (field). A field is a collection of data elements with the same data type.

FIR Filter

(Finite Impulse Response Filter) a filter with finite impulse answer

Formal parameters

Input / Output parameters, which are used within the logic of a FFB and led out of the FFB as inputs/outputs.

Function (FUNC)

A program organization unit, which supplies an exact data element when processing. a function has no internal status information. Multiple invocations of the same function using the same input parameters always supply the same output values.

Details of the graphic form of the function invocations can be found in the definition "Functional block (instance)". In contrast to the invocations of the function blocks, function invocations only have a single unnamed output, whose name is the same as the function. In FBD each invocation is denoted by a unique number via the graphic block, this number is automatically generated and can not be altered.

Function block (Instance) (FB)

A function block is a program organization unit, which correspondingly calculates the functionality values that were defined in the function block type description, for the outputs and internal variable(s), if it is invoked as a certain instance. All internal variable and output values for a certain function block instance remain from one function block invocation to the next. Multiple invocations of the same function block instance with the same arguments (input parameter values) do not therefore necessarily supply the same output value(s).

Each function block instance is displayed graphically using a rectangular block symbol. The name of the function block type is stated in the top center of the rectangle. The name of the function block instance is also stated at the top, but outside of the rectangle. It is automatically generated when creating an instance, but, depending on the user's requirements, it can be altered by the user. Inputs are displayed on the left side of the block and outputs are displayed on the right side. The names of the formal input/output parameters are shown inside the rectangle in the corresponding places.

The above description of the graphic display is especially applicable to the function invocations and to DFB invocations. Differences are outlined in the corresponding definitions.

Function Block Dialog (FBD)

One or more sections, which contain graphically displayed networks from Functions, Function blocks and Connections.

Function block type

A language element, consisting of: 1. the definition of a data structure, divided into input, output and internal variables; 2. a set of operations, which are performed with elements of the data structure, when a function block type instance is invoked. This set of operations can either be formulated in one of the IEC languages (DFB type) or in "C" (EFB type). A function block type can be instanced (invoked) several times.

Function Number

The function number is used to uniquely denote a function in a program or DFB. The function number can not be edited and is automatically assigned. The function number is always formed as follows: .n.m

n = Number of the section (consecutive numbers)

m = Number of the FFB object in the section (current number)

G

Generic Data Type

A data type, which stands in place of several other data types.

Generic literals

If the literal's data type is not relevant, simply specify the value for the literal. If this is the case, Concept automatically assigns the literal a suitable data type.

Global Data

Global data are Unlocated variables.

Global derived data types

Global derived data types are available in each Concept project and are occupied in the DFB directory directly under the Concept directory.

Global DFBs

Global DFBs are available in each Concept project. The storage of the global DFBs is dependant upon the settings in the CONCEPT.INI file.

Global macros

Global macros are available in each Concept project and are stored in the DFB directory directly under the Concept directory.

Groups (EFBs)

Some EFB libraries (e.g. the IEC library) are divided into groups. This facilitates locating the EFBs especially in expansive libraries.

H

Host Computer

Hardware and software, which support programming, configuring, testing, operating and error searching in the PLC application as well as in a remote system application, in order to enable source documentation and archiving. The programming device can also be possibly used for the display of the process.

I

I/O Map

The I/O and expert modules from the various CPUs are configured in the I/O map.

Icon

Graphical representation of different objects in Windows, e.g. drives, application programs and document windows.

IEC 61131-3

International standard: Programmable Logic Controls - Part 3: Programming languages.

IEC Format (QW1)

There is an IEC type designation in initial position of the address, followed by the five-figure address.

- %0x12345 = %Q12345
- %1x12345 = %I12345
- %3x12345 = %IW12345
- %4x12345 = %QW12345

IEC name conventions (identifier)

An identifier is a sequence of letters, numbers and underscores, which must begin with either a letter or underscore (i.e. the name of a function block type, an instance, a variable or a section). Letters of a national typeface (i.e.: ö,ü, é, õ) can be used, except in project and DFB names.

Underscores are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as two separate identifiers. Several leading and multiple successive underscores are not allowed.

Identifiers should not contain any spaces. No differentiation is made between upper and lower case, e.g. "ABCD" and "abcd" are interpreted as the same identifier.

Identifiers should not be Keywords.

IEC Program Memory

The IEC program memory consists of the program code, EFB code, the section data and the DFB instance data.

IIR Filter

(Infinite Impulse Response Filter) a filter with infinite impulse answer

Initial step

The first step in a sequence. A step must be defined as an initial step for each sequence. The sequence is started with the initial step when first invoked.

Initial value

The value, which is allocated to a variable when the program is started. The values are assigned in the form of literals.

Input bits (1x references)

The 1/0 status of the input bits is controlled via the process data, which reaches from an input device to the CPU.

NOTE: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 100201 signifies an output or marker bit at the address 201 in the State RAM.

Input parameter (Input)

Upon invocation of a FFB, this transfers the corresponding argument.

Input words (3x references)

An input word contains information, which originates from an external source and is represented by a 16 bit number. A 3x register can also contain 16 sequential input bits, which were read into the register in binary or BCD (binary coded decimal) format. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 300201 signifies a 16-bit input word at the address 201 in the State RAM.

Instance Name

An identifier, which belongs to a certain function block instance. The instance name is used to clearly denote a function block within a program organization unit. The instance name is automatically generated, but it can be edited. The instance name must be unique throughout the whole program organization unit, and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must comply with the IEC name conventions otherwise an error message appears. The automatically generated instance name is always formed as follows: FBI_n_m

FBI = Function Block Instance

n = Number of the section (consecutive numbers)

m = Number of the FFB object in the section (current number)

Instancing

Generating an Instance.

Instruction (IL)

Instructions are the "commands" of the IL programming language. Each instruction begins on a new line and is performed by an operator with a modifier if necessary, and if required for the current operation, by one or more operands. If several operands are used, they are separated by commas. A character can come before the instruction, which is then followed by a colon. The comment must, if present, be the last element of the line.

Instruction (LL984)

When programming electrical controls, the user must implement operation-coded instructions in the form of picture objects, which are divided into a recognizable contact form. The designed program objects are, on a user level, converted to computer usable OP codes during the download process. The OP codes are decoded in the CPU and processed by the firmware functions of the controller in a way that the required control is implemented.

Instruction (ST)

Instructions are "commands" of the ST programming language. Instructions must be completed by semicolons. Several instructions can be entered in one line (separated by semicolons).

Instruction list (IL)

IL is a text language according to IEC 1131, which is shown in operations, i.e. conditional or unconditional invocations of Functions blocks and Functions, conditional or unconditional jumps etc. through instructions.

INT

INT stands for the data type "whole number (integer)". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 16 bits. The value range for variables of this datatype reaches from -2 exp (15) to 2 exp (15) -1.

Integer literals

Integer literals are used to input whole number values into the decimal system. The values can have a preceding sign (+/-). Single underscores (_) between numbers are not significant.

Example

-12, 0, 123_456, +986

INTERBUS (PCP)

The new INTERBUS (PCP) I/O drop type is entered into the Concept configurator, to allow use of the INTERBUS PCP channel and the INTERBUS process data preprocessing (PDV). This I/O drop type is assigned the INTERBUS switching module 180-CRP-660-01.

The 180-CRP-660-01 differs from the 180-CRP-660-00 only in the fact that it has a clearly larger I/O range in the control state RAM.

Invocation

The process by which the execution of an operation is initiated.

J

Jump

Element of the SFC language. Jumps are used to skip zones in the sequence.

K

Keywords

Keywords are unique combinations of characters, which are used as special syntactical components, as defined in Appendix B of the IEC 1131-3. All keywords which are used in the IEC 1131-3 and therefore in Concept, are listed in Appendix C of the IEC 1131-3. These keywords may not be used for any other purpose, i.e. not as variable names, section names, instance names etc.

L

Ladder Diagram (LD)

Ladder Diagram is a graphic programming dialog according to IEC1131, which is optically oriented to the "rung" of a relay contact plan.

Ladder Logic 984 (LL)

The terms Ladder Logic and Ladder Diagram refer to the word Ladder being executed. In contrast to a circuit diagram, a ladder diagram is used by electrotechnicians to display an electrical circuit (using electrical symbols), which should show the course of events and not the existing wires, which connect the parts with each other. A usual user interface for controlling the actions of automation devices permits a Ladder Diagram interface, so that electrotechnicians do not have to learn new programming languages to be able to implement a control program.

The structure of the actual Ladder Diagram enables the connection of electric elements in such a way that generates a control output, which is dependent upon a logical power flow through used electrical objects, which displays the previously requested condition of a physical electrical device.

In simple form, the user interface is a video display processed by the PLC programming application, which sets up a vertical and horizontal grid in which programming objects are classified. The diagram contains the power grid on the left side, and when connected to activated objects, the power shifts from left to right.

Landscape

Landscape means that when looking at the printed text, the page is wider than it is high.

Language Element

Every basic element in one of the IEC programming languages, e.g. a step in SFC, a function block instance in FBD or the initial value of a variable.

Library

Collection of software objects, which are intended for re-use when programming new projects, or even building new libraries. Examples are the libraries of the Elementary function block types.

EFB libraries can be divided up into Groups.

Link

A control or data flow connection between graphical objects (e.g. steps in the SFC Editor, function blocks in the FBD Editor) within a section, represented graphically as a line.

Literals

Literals are used to provide FFB inputs, and transition conditions etc with direct values. These values can not be overwritten by the program logic (write-protected). A distinction is made between generic and standardized literals.

Literals are also used to allocate, to a constant, a value or a variable, an initial value.

Entries are made as base 2 literal, base 8 literal, base 16 literal, integer literal, real literal or real literal with exponent.

Local derived data types

Local derived data types are only available in a single Concept project and the local DFBs and are placed in the DFB directory under the project directory.

Local DFBs

Local DFBs are only available in a single Concept project and are placed in the DFB directory under the project directory.

Local Link

The local network is the network, which connects the local nodes with other nodes either directly or through bus repeaters.

Local macros

Local macros are only available in a single Concept project and are placed in the DFB directory under the project directory.

Local network nodes

The local node is the one which is currently being configured.

Located variable

A state RAM address (reference addresses 0x, 1x, 3x,4x) is allocated to located variables. The value of these variables is saved in the state RAM and can be modified online using the reference data editor. These variables can be addressed using their symbolic names or their reference addresses.

All inputs and outputs of the PLC are connected to the state RAM. The program can only access peripheral signals attached to the PLC via located variables. External access via Modbus or Modbus Plus interfaces of the PLC, e.g. from visualization systems, is also possible via located variables.

M

Macro

Macros are created with the help of the Concept DFB software.

Macros are used to duplicate frequently used sections and networks (including their logic, variables and variable declaration).

A distinction is made between local and global macros.

Macros have the following properties:

- Macros can only be created in the FBD and LD programming languages.
- Macros only contain one section.
- Macros can contain a section of any complexity.
- In programming terms, there is no difference between an instanced macro, i.e. a macro inserted into a section and a conventionally created section.
- DFB invocation in a macro
- Declaring variables
- Using macro-specific data structures
- Automatic transfer of the variables declared in the macro.
- Initial values for variables
- Multiple instancing of a macro in the entire program with differing variables
- The name of the section, variable names and data structure names can contain up to 10 different exchange marks (@0 to @9).

MMI

Man-Machine-Interface

Multi element variables

Variables to which a Derived data type defined with STRUCT or ARRAY is allocated.

A distinction is made here between field variables and structured variables.

N

Network

A network is the collective switching of devices to a common data path, which then communicate with each other using a common protocol.

Network node

A node is a device with an address (1...64) on the Modbus Plus network.

Node

Node is a programming cell in a LL984 network. A cell/node consists of a 7x11 matrix, i.e. 7 rows of 11 elements.

Node Address

The node address is used to uniquely denote a network node in the routing path. The address is set on the node directly, e.g. using the rotary switch on the back of the modules.

O

Operand

An operand is a literal, a variable, a function invocation or an expression.

Operator

An operator is a symbol for an arithmetic or boolean operation which is to be executed.

Output parameter (output):

A parameter, through which the result(s) of the evaluation of a FFB is/are returned.

Output/Marker bits (0x references)

An output/marker bit can be used to control real output data using an output unit of the control system, or to define one or more discrete outputs in the state RAM. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 000201 signifies an output or marker bit at the address 201 in the State RAM.

Output/marker words (4x references)

An output / marker word can be used to save numerical data (binary or decimal) in the state RAM, or to send data from the CPU to an output unit in the control system. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 400201 signifies a 16 bit output or marker word at the address 201 in the State RAM.

P

Peer CPU

The Peer CPU processes the token execution and the data flow between the Modbus Plus network and the PLC user logic.

PLC

Memory programmable controller

Portrait

Portrait means that the sides are larger than the width when printed.

Program

The uppermost program organization unit. A program is closed on a single PLC download.

Program organization unit

A function, a function block, or a Program. This term can refer to either a type or an instance.

Program redundancy system (Hot Standby)

A redundancy system consists of two identically configured PLC machines, which communicate with one another via redundancy processors. In the case of a breakdown of the primary PLC, the secondary PLC takes over the control check. Under normal conditions, the secondary PLC does not take over the control function, but checks the status information, in order to detect errors.

Project

General description for the highest level of a software tree structure, which specifies the super-ordinate project name of a PLC application. After specifying the project name you can save your system configuration and your control program under this name. All data that is created whilst setting up the configuration and program, belongs to this super-ordinate project for this specific automation task.

General description for the complete set of programming and configuration information in the project database, which represents the source code that describes the automation of a system.

Project database

The database in the host computer, which contains the configuration information for a project.

Prototype file (Concept-EFB)

The prototype file contains all the prototypes of the assigned functions. In addition, if one exists, a type definition of the internal status structure is specified.

R

REAL

REAL stands for the data type "floating point number". The entry can be real-literal or real-literal with an exponent. The length of the data element is 32 bits. The value range for variables of this data type extends from +/-3.402823E+38.

NOTE: Dependent on the mathematical processor type of the CPU, different ranges within this permissible value range cannot be represented. This applies to values that are approaching ZERO and for values that approach INFINITY. In these cases NAN (**N**ot **A N**umber) or INF (**INF**inite) will be displayed in the animation mode instead of a number value.

Real literals

Real literals are used to input floating point values into the decimal system. Real literals are denoted by a decimal point. The values can have a preceding sign (+/-). Single underscores (_) between numbers are not significant.

Example

-12.0, 0.0, +0.456, 3.14159_26

Real literals with exponents

Real literals with exponents are used to input floating point values into the decimal system. Real literals with exponents are identifiable by a decimal point. The exponent indicates the power of ten, with which the existing number needs to be multiplied in order to obtain the value to be represented. The base can have a preceding negative sign (-). The exponent can have a preceding positive or negative sign (+/-). Single underscores (_) between numbers are not significant. (Only between characters, not before or after the decimal point and not before or after "E", "E+" or "E-")

Example

-1.34E-12 or -1.34e-12

1.0E+6 or 1.0e+6

1.234E6 or 1.234e6

Reference

Every direct address is a reference that begins with an indicator, which specifies whether it is an input or an output and whether it is a bit or a word. References that begin with the code 6, represent registers in the extended memory of the state RAM.

0x range = Output/Marker bits

1x range = Input bits

3x range = Input words

4x range = Output registers

6x range = Register in the extended memory

NOTE: The x, which follows each initial reference type number, represents a five-digit storage location in the user data memory, i.e. the reference 400201 signifies a 16 bit output or marker word at the address 201 in the State RAM.

Register in the extended memory (6x-reference)

6x references are holding registers in the extended memory of the PLC. They can only be used with LL984 user programs and only with a CPU 213 04 or CPU 424 02.

Remote Network (DIO)

Remote programming in the Modbus Plus network enables maximum performance when transferring data and dispenses with the need for connections. Programming a remote network is simple. Setting up a network does not require any additional ladder logic to be created. All requirements for data transfer are fulfilled via corresponding entries in the Peer Cop Processor.

RIO (Remote I/O)

Remote I/O indicates a physical location of the I/O point controlling devices with regard to the CPU controlling them. Remote inp./outputs are connected to the controlling device via a twisted communication cable.

RTU-Mode

Remote Terminal Unit

The RTU mode is used for communication between the PLC and an IBM compatible personal computer. RTU works with 8 data bits.

Runtime error

Errors, which appear during program processing on the PLC, in SFC objects (e.g. Steps) or FFBs. These are, for example, value range overflows for numbers or timing errors for steps.

S

SA85 module

The SA85 module is a Modbus Plus adapter for IBM-AT or compatible computers.

Scan

A scan consists of reading the inputs, processing the program logic and outputting the outputs.

Section

A section can for example be used to describe the functioning mode of a technological unit such as a motor.

A program or DFB consists of one or more sections. Sections can be programmed with the IEC programming languages FBD and SFC. Only one of the named programming languages may be used within a section at any one time.

Each section has its own document window in Concept. For reasons of clarity, however, it is useful to divide a very large section into several small ones. The scroll bar is used for scrolling within a section.

Section Code

Section Code is the executable code of a section. The size of the Section Code is mainly dependent upon the number of blocks in the section.

Section Data

Section data is the local data in a section such as e.g. literals, connections between blocks, non-connected block inputs and outputs, internal status memory of EFBs.

NOTE: Data which appears in the DFBs of this section is not section data.

Separator Format (4:00001)

The first digit (the reference) is separated from the five-digit address that follows by a colon (:).

Sequence language (SFC)

The SFC Language Elements enable a PLC program organization unit to be divided up into a number of Steps and Transitions, which are connected using directional Links. A number of actions belong to each step, and transition conditions are attached to each transition.

Serial Connections

With serial connections (COM) the information is transferred bit by bit.

Source code file (Concept-EFB)

The source code file is a normal C++ source file. After executing the **Library** → **Create files** menu command, this file contains an EFB-code frame, in which you have to enter a specific code for the EFB selected. To do this invoke the **Objects** → **Source** menu command.

Standard Format (400001)

The five-digit address comes directly after the first digit (the reference).

Standardized literals

If you would like to manually determine a literal's data type, this may be done using the following construction: ’Data type name’#’value of the literal’.

Example

INT#15 (Data type: integer, value: 15),

BYTE#00001111 (Data type: byte, value: 00001111)

REAL#23.0 (Data type: real, value: 23.0)

To assign the data type REAL, the value may also be specified in the following manner: 23.0.

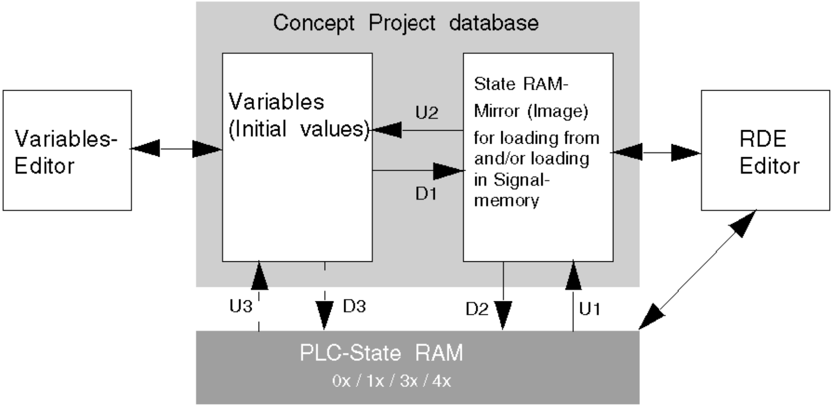
Entering a comma will automatically assign the data type REAL.

State RAM

The state RAM is the memory space for all variables, which are accessed via References (Direct representation) in the user program. For example, discrete inputs, coils, input registers, and output registers are located in the state RAM.

State RAM overview for uploading and downloading

Overview:


Concept Project database
Variables-Editor
Variables (Initial values)
U2
D1
State RAM-Mirror (Image) for loading from and/or loading in Signal-memory
RDE Editor
U3
D3
D2
U1
PLC-State RAM
0x / 1x / 3x / 4x

Status Bits

For every device with global inputs or specific inputs/outputs of Peer Cop data, there is a status bit. If a defined group of data has been successfully transferred within the timeout that has been set, the corresponding status bit is set to 1. If this is not the case, this bit is set to 0 and all the data belonging to this group is deleted (to 0).

Step

SFC-language element: Situation, in which the behavior of a program, in reference to its inputs and outputs, follows those operations which are defined by the actions belonging to the step.

Step name

The step name is used to uniquely denote a step in a program organization unit. The step name is generated automatically, but it can be edited. The step name must be unique within the entire program organization unit, otherwise an error message will appear.

The automatically generated step name is always formed as follows: S_n_m

S = step

n = Number of the section (consecutive numbers)

m = Number of the step in the section (current number)

Structured text (ST)

ST is a text language according to IEC 1131, in which operations, e.g. invocations of Function blocks and Functions, conditional execution of instructions, repetitions of instructions etc. are represented by instructions.

Structured variables

Variables to which a Derived data type defined with STRUCT (structure) is allocated.

A structure is a collection of data elements with generally different data types (elementary data types and/or derived data types).

SY/MAX

In Quantum control devices, Concept includes the preparation of I/O-map SY/MAX-I/O modules for remote controlling by the Quantum PLC. The SY/MAX remote backplane has a remote I/O adapter in slot 1, which communicates via a Modicon S908 R I/O System. The SY/MAX-I/O modules are executed for you for labeling and inclusion in the I/O map of the Concept configuration.

T

Template file (Concept-EFB)

The template file is an ASCII file with layout information for the Concept FBD Editor, and the parameters for code creation.

TIME

TIME stands for the data type "time". The entry is time literal. The length of the data element is 32 bits. The value range for variables of this data type extends from 0 to 2exp(32)-1. The unit for the data type TIME is 1 ms.

Time literals

Permissible units for times (TIME) are days (D), hours (H), minutes (M), seconds (S) and milliseconds (MS) or combinations of these. The time must be marked with the prefix t#, T#, time# or TIME#. The "overflow" of the unit with the highest value is permissible, e.g. the entry T#25H15M is allowed.

Example

t#14MS, T#14.7S, time#18M, TIME#19.9H, t#20.4D, T#25H15M, time#5D14H12M18S3.5MS

Token

The network "token" controls the temporary possession of the transfer right via a single node. The token passes round the nodes in a rotating (increasing) address sequence. All nodes follow the token rotation and can receive all the possible data that is sent with it.

Total IEC memory

The total IEC memory consists of the IEC program memory and the global data.

Traffic Cop

The traffic cop is an IO map, which is generated from the user-IO map. The traffic cop is managed in the PLC and in addition to the user IO map, contains e.g. status information on the I/O stations and modules.

Transition

The condition, in which the control of one or more predecessor steps passes to one or more successor steps along a directed link.

U

UDEFB

User-defined elementary functions/function blocks

Functions or function blocks, which were created in the C programming language, and which Concept provides in libraries.

UDINT

UDINT stands for the data type "unsigned double integer". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 32 bits. The value range for variables of this data type extends from 0 to 2exp(32)-1.

UINT

UINT stands for the data type "unsigned integer". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 16 bits. The value range for variables of this data type extends from 0 to (2exp 16)-1.

Unlocated variable

Unlocated variables are not allocated a state RAM address. They therefore do not occupy any state RAM addresses. The value of these variables is saved in the internal system and can be changed using the reference data editor. These variables are only addressed using their symbolic names.

Signals requiring no peripheral access, e.g. intermediate results, system tags etc., should be primarily declared as unlocated variables.

V

Variables

Variables are used to exchange data within a section, between several sections and between the program and the PLC.

Variables consist of at least one variable name and one data type.

If a variable is assigned a direct address (reference), it is called a located variable. If the variable has no direct address assigned to it, it is called an unlocated variable. If the variable is assigned with a derived data type, it is called a multi element variable.

There are also constants and literals.

W

Warning

If a critical status is detected during the processing of a FFB or a step (e.g. critical input values or an exceeded time limit), a warning appears, which can be seen using the **Online →Event Viewer...**menu command. For FFBs, the ENO remains set to "1".

WORD

WORD stands for the data type "bit sequence 16". Entries are made as base 2 literal, base 8 literal or base 16 literal. The length of the data element is 16 bits. A numerical value range can not be assigned to this data type.

Index

Symbols

- Modbus Presettings
  - Interface Settings in Windows NT, *1028*
- PLC Configuration
  - Export, *728*
  - Import, *728*
- Profibus DP controller, *926*
- program creation
  - LD, *267*
- ’SFCSTEP_STATE’ variable, *277*
- ’SFCSTEP_TIMES’ variable, *276*
- ’step’-variable, *277*
- •General information about online functions, *632*
- => Assignment, *376*, *445*

A

- Access Right, *767*
- Access Rights, *760*, *769*
- Action, *278*
- Action variable, *278*
- Actions
  - Process, *299*
- activate dialogs, *124*
- Actual parameters
  - FBD, *221*
- actual parameters
  - LD, *254*
- alias designations
  - step, *307*
  - transition, *307*
- alternative branch, *287*
- Alternative connection, *289*
- Animation, *606*, *746*, *748*
  - FBD, *233*
  - General information, *681*
  - IEC section, *682*
  - IL, *389*
  - IL/ST, *386*
- animation
  - LD, *265*
- Animation
  - LL984 section, *684*
  - Section, *681*
- animation
  - SFC, *311*, *313*
- ANY Outputs, *442*
- Archiving
  - DFB, *741*
  - EFB, *741*
  - Project, *741*
- ARRAY
  - Range Monitoring, *590*
- ASCII message editor, *611*, *612*, *618*
  - Combination mode, *629*, *629*
  - Control code, *616*
  - Direct mode, *629*, *629*
  - Flush (buffer), *619*
  - Generals, *613*
  - How to continue after getting a warning,

628, *628*
How to Use, *622*
Message Number, *624*
Message text, *626*
Offline mode, *629*, *629*
Repeat, *620*
Simulation text, *627*
Spaces, *617*
Text, *614*
User interface, *621*, *622*
Variables, *615*
ASCII messages, *83*, *123*
Assign instructions
ST, *422*
Assignment
=>, *445*
Atrium
Memory optimization, *204*
Atrium example
INTERBUS controller, *955*
Atrium first startup
DOS Loader, *1092*
EXECLoader, *1072*
Modbus Plus, *1072*, *1092*
Atrium INTERBUS controller, *956*
Auto-Log-Out, *151*
Automatic Connection, *1148*, *1151*
available functions in OFFLINE and ONLINE modes, *105*

B

Backplane Expander
Edit I/O Map, *135*
Error handling, *136*
Generals, *134*
Backplane Expander Config
Configure, *133*
Block call up
IL, *371*
ST, *441*

C

call
DFB, *368*
FFB, *368*
Call
FFB, *379*
Project, *811*
Chain jump, *285*
Chain loop, *286*
change
coil, LD, *259*
contact, LD, *259*
FFB, FBD, *226*
FFB, LD, *259*
Changing signal states of a Located variable
Reference data editor, *600*
Close Column
LL984, *465*
Closer
LD, *243*
code generation
FBD, *232*
Code generation
IL, *384*
code generation
LD, *264*
Code generation
ST, *450*
coil
change, LD, *259*
replace, LD, *259*
Coil - negated
LD, *246*
Coil - reset
LD, *247*
Coil - set
LD, *247*
Coil – negative edge
LD, *246*
Coil – positive edge
LD, *246*
Coils
LD, *245*
Cold restart, *65*

Comments
    Data type editor, *585*
    Derived data type, *585*
Communication, *38*
Compact
    Memory optimization, *187*
Compact configuration
    RTU extension, *143*
Compact example, *950*
Compact first startup
    DOS Loader, *1053*, *1089*
    EXECLoader, *1036*, *1068*
    Modbus, *1036*, *1053*
    Modbus Plus, *1068*, *1089*
Concept DFB, *481*, *521*
Concept M
    Hardware Package Contents, *28*
Concept ModConnect, *995*
    Integrating new Modules, *998*
    Removing modules, *999*
    Use of Third Party Modules in Concept, *1000*
Concept PLCSIM32, *748*
Concept S
    Hardware Package Contents, *28*
Concept Security, *757*, *758*, *760*, *767*, *769*, *770*
Concept SIM, *746*
Concept XL
    Hardware Package Contents, *28*
CONCEPT.INI, *1105*, *1106*
    Exclusion of the global/local DFBs from Online-Backup, *1110*
    general, *1107*
    LD section settings, *1114*
    Path for global DFBs, *1110*
    Path for Help Files, *1110*
    print settings, *1108*
    Project Name Definition, *1109*
    Reading global DFBs, *1110*
    Register Address Format Settings, *1109*
    representation of internal data, *1113*
    Saving the global DFBs during upload, *1110*
    Security Settings, *1117*
    Setting for Online Processing, *1115*
    Setting for the Address Format, *1116*
    Settings for the RDE, *1118*
    Settings for the Tools menu, *1119*
    Settings for Warning Messages, *1116*
    Variable Storage Settings, *1109*
Configuration, *99*
    General information, *100*
    Optional, *122*
    Unconditional, *107*
Configuration example
    Atrium-INTERBUS controller, *955*
    Compact controller, *950*
    Momentum-Ethernet bus system, *974*
    Momentum-Remote I/O bus, *965*
    Quantum-INTERBUS control, *910*
    Quantum-Peer Cop, *941*
    Quantum-Profibus DP controller, *925*
    Quantum-Remote control with DIO, *900*
    Quantum-Remote control with RIO, *878*
    Quantum-Remote control with RIO (series 800), *887*
    Quantum-SY/MAX controller, *916*
configuration extensions, *124*
Configuration in OFFLINE and ONLINE mode, *103*
    General information, *104*
Configuration of various network systems, *137*
Configurator
    Ethernet I/O Scanner, *144*
configure, *79*
Configure
    Backplane Expander Config, *133*
configure
    Ethernet, *141*
Configure
    INTERBUS, *138*
configure
    Profibus DP, *139*
Configure
    RTU extension, *143*
configure Ethernet, *141*

Configure INTERBUS system, *138*
configure network systems, *124*
Configure network systems, *137*
configure Profibus DP system, *139*
Connect
  PLC, *633*
Connect to PLC, *633*
connecting IEC Simulator (32-bit), *647*
connecting to IEC Simulator (32-bit), *647*
Connecting to the PLC
  automatically with command line parameters, *1148*
  Automatically with the CCLaunch Tool, *1151*
Constant Scan, *651*
Constants, *64*
contact
  change, LD, *259*
  replace, LD, *259*
Contacts
  LD, *243*, *243*, *244*
Context help, *813*
Convert
  DFBs, *991*
  Macros, *991*
  Projects, *991*
converting RDE templates, *598*
Convertion
  Modsoft programs, *1001*
CPU selection for the PLC type, *110*
Create
  DFB, *501*
create
  FFB, FBD, *225*
  FFB, LD, *258*
Create
  Macro, *531*
  Program, *75*
  Project, *75*
  Project Symbol, *811*
Creating a program
  IL, *390*
Cyclical Setting of Variables
  Reference Data Editor, *602*

D

data exchange between nodes on the Modbus Plus network, *125*
data flow, *262*
  FBD, *230*
Data Protection, *82*
data protection in the state RAM, *128*
Data Type Definition
  Extended (larger than 64 Kbytes), *572*
data type editor, *567*
Data type editor, *565*, *566*
  Comments, *585*
  Elements, *575*
data type editor
  key words, *577*
  separators, *583*
Data Type Editor
  Short Cut Keys, *834*
Data type editor
  Syntax, *574*
data type editor
  use of memory, *586*
Datatype editor
  Names, *582*
DDT, *572*
Declaration of variables, *545*
Declare
  Actions, *299*
  Step properties, *297*
declare
  transition, *305*
Declare Variables, *545*
Defining Colors
  INI File, *1115*
defining the LD contact connection
  settings in the INI file, *1114*
defining the number of LD columns/fields
  settings in the INI file, *1114*
Delete
  DFB, *743*
  Macro, *743*
  Project, *743*
Deleting memory zones from the PLC, *654*
Deleting PLC contents, *654*
derived data type, *567*

Derived data type, *565*
Derived Data Type, *566*
Derived data type
    Comments, *585*
Derived Data Type
    Elements, *575*
    Export, *697*
derived data type
    global, *570*
    key words, *577*
    local, *570*
Derived data type
    Names, *582*
derived data type
    separators, *583*
Derived Data Type
    Syntax, *574*
derived data type
    use of memory, *586*
Derived Data Types
    Use, *588*
Derived Function Block, *483*
derived function block
    LD, *250*
Derived Function Blocks
    FBD, *218*
DFB, *481*, *483*
    Archiving, *741*
    call, *368*
    context sensitive help, *499*
    Create, *501*
    Creating Global Variables, *496*
    Delete, *743*
    Documentation, *730*
    FBD, *218*
    global, *485*
    invocation, *372*, *442*
    LD, *250*, *250*
    local, *485*
    Protect, *770*
DFBs
    Convert, *991*
Diagnosis
    Transition diagnosis, *319*
diagnosis viewer, *685*
diagnostics viewer, *685*
Dialog boxes, *808*
Dialog interaction
    LL984, *459*
Direct Addresses, *64*
Disable
    Interrupt Sections, *70*
    Section, *69*
document section options, *735*
Documentation
    Contents, *731*
    DFB, *730*
    Keywords, *739*
documentation
    layout, *732*
Documentation
    Macro, *730*
    Project, *730*
DOS Loader
    Atrium first startup, *1092*
    Compact first startup, *1053*, *1089*
    Momentum first startup, *1056*, *1059*, *1095*, *1098*
    Quantum first startup, *1050*, *1086*
    Startup when using Modbus, *1049*
    Startup when using Modbus Plus, *1085*
Download Changes, *674*
Downloading Changes, *674*
Driver for 16 bit application capability with Windows 98/2000/NT
    Virtual MBX Driver, *1019*
Driver for connection between ModConnect Host interface adapters and 32 bit applications with Windows 98/2000/NT
    MBX-Treiber, *1020*
Driver for Modbus Plus Function via TCP/IP
    Ethernet MBX Driver, *1022*
Driver for Remote Operation
    Remote MBX Driver, *1021*
DTY, *565*, *566*, *567*
DX Zoom
    LL984, *466*

E

Edit
    Actions, *299*
    LL984, *459*, *464*
edit
    SFC, *293*
Edit
    SFC, *294*
    Step properties, *297*
edit
    transition, *305*
Edit I/O Map
    Backplane Expander, *135*
Editing local Drop, *879*
Editing Networks
    LL984, *465*
Editors, *33*
EFB
    Archiving, *741*
    FBD, *216*
    LD, *248*
EFBs for Interrupt Sections, *1146*
Elementary Function
    FBD, *216*
elementary function
    LD, *248*
Elementary Function Block
    FBD, *217*
elementary function block
    LD, *249*
Elements
    Data type editor, *575*
    Derived Data Type, *575*
EN
    FBD, *219*
    LD, *252*
ENC File, *39*, *687*, *687*
Encoded Log, *39*
Encrypt Logfile, *759*
Encrypted Logging
    ENC File, *687*
ENO
    FBD, *219*
    LD, *252*
EQUAL, *634*
Equation network
    LL984, *471*, *472*
Equation network, Syntax and Semantics
    LL984, *476*
Error handling
    Backplane Expander, *136*
Establishing the hardware connection
    Modbus Plus presettings, *1024*
    Modbus presettings, *1029*
Ethernet, *646*
Ethernet / I/O Scanner
    How to use the Ethernet / I/O Scanner, *149*
Ethernet Bus System
    Create online connection, *990*
Ethernet Bus System (Momentum), *975*
Ethernet I/O Scanner
    Configurator, *144*
Ethernet MBX Driver
    Driver for Modbus Plus Function via TCP/IP, *1022*
ethernet with Momentum, *142*
Ethernet with Quantum, *141*
Event Viewer
    INI Settings, *1122*
Example of hardware configuration
    Atrium-INTERBUS controller, *955*
    Compact controller, *950*
    Momentum-Ethernet bus system, *974*
    Momentum-Remote I/O bus, *965*
    Quantum-INTERBUS control, *910*
    Quantum-Peer Cop, *941*
    Quantum-Profibus DP controller, *925*
    Quantum-Remote control with DIO, *900*
    Quantum-Remote control with RIO, *878*
    Quantum-Remote control with RIO (Series 800), *887*
    Quantum-SY/MAX controller, *916*
Exchange Marking
    Macro, *527*
Exclusion of the global/local DFBs from Online-Backup
    Settings in the INI-File, *1110*

- EXEC file
  - CPU 424 02, *165*
  - CPU X13 0X, *165*
  - Momentum, *200*
- EXEC files, *1101*
- EXECLoader
  - Atrium first startup, *1072*
  - Compact first startup, *1036*, *1068*
  - Momentum first startup, *1040*, *1044*, *1076*, *1080*
  - Quantum first startup, *1032*, *1064*
  - Startup when using Modbus, *1031*
  - Startup when using Modbus Plus, *1063*
- execution order
  - FBD, *227*
- Execution order
  - Section, *69*
- Execution Order
  - Timer Event Sections, *1134*
- Execution sequence
  - LD, *260*
- Export, *691*
  - Derived Data Type, *697*
  - General Information, *692*
- export
  - PLC configuration, *727*
- Export
  - PLC Configuration, *728*
  - Section, *694*
  - Variable, *697*
- Exporting located variables, *556*
- Expressions
  - ST, *396*
- extended memory, *169*

F

- Factory Link, *724*
- FBD, *213*
  - Actual parameters, *221*
  - Animation, *233*
  - Calling a macro, *540*
  - code generation, *232*
  - data flow, *227*, *230*
  - Derived Function Blocks, *218*
  - DFB, *218*
  - EFB, *216*
  - Elementary Function, *216*
  - Elementary Function Block, *217*
  - EN, *219*
  - ENO, *219*
  - execution order, *227*
  - FFB, *216*
  - Function, *216*
  - Function Block, *217*
  - Icon bar, *822*
  - link, *220*
  - loop, *230*
  - Online Functions, *233*
  - program creation, *235*
  - Short Cut Keys, *837*
  - Text Object, *223*
  - UDEFB, *219*
  - User-defined Elementary Function, *219*
  - User-defined Elementary Function Block, *219*
- FFB
  - call, *368*
  - Call, *379*
  - change, FBD, *226*
  - change, LD, *259*
  - create, FBD, *225*
  - create, LD, *258*, *258*
  - FBD, *216*
  - insert, FBD, *225*
  - insert, LD, *258*
  - invocation, *372*, *442*, *446*
  - LD, *248*
  - position, *225*, *258*
  - replace, FBD, *226*
  - replace, LD, *259*
- forcing
  - I/O, *596*

Function
FBD, *216*
function
LD, *248*
Function Block
FBD, *217*
function block
LD, *249*
Function Block language, *213*
Function Blocks for Interrupt Sections, *1146*

G

General, *23*
PLC Connection, *634*
General information
Loading a project, *671*
Online functions, *632*
Select process information, *664*
General information about configuration in OFFLINE and ONLINE mode, *104*
General information about hardware configuration, *100*
general information about the online control panel, *650*
General information about the PLC configuration, *101*
General Information about the Reference Data Editor, *596*
General to the variables editor , *544*
Generals
Backplane Expander, *134*
Generate
Project symbol, *811*
Global data transfer
Peer Cop, *946*
global derived data type, *570*
global DFB, *485*
Global DFBs
Defining the Path, *1110*
INI File, *1110*
Reading, *1112*
Storing, *1111*
global macro, *525*
Global Variables in DFBs, *496*

H

hardware
performance, *775*
Head setup, *81*
Help, *813*
Help Files
Defining the Path, *1110*
How to use the Ethernet / I/O Scanner
Ethernet / I/O Scanner, *149*

I

I/O
forcing, *596*
I/O Event Section
Handling, *1125*
I/O Event Sections, *1142*
Priority, *1143*
Runtime Error, *1144*
I/O map, *80*, *119*
Icon bar, *820*, *821*, *822*, *823*, *825*
icons, *829*, *829*, *830*, *830*
Icons, *819*, *820*, *820*, *821*, *821*, *822*, *822*, *823*, *823*, *825*, *825*, *827*, *828*, *831*, *831*
Icons_Project Browser, *831*
Identifier, *302*
IEC
Momentum first startup, *1040*, *1076*, *1095*
IEC conformity, *849*
IEC Hot Standby data, *113*
IEC section
Animation, *682*

IL, *321*
- Animation, *386*, *389*
- Block call up, *371*
- Code generation, *384*
- Creating a program, *390*
- Instruction, *324*, *325*
- List of Symbols, *827*
- Modifier, *329*
- Online functionen, *389*
- Online functions, *385*, *386*
- Operands, *327*
- Operators, *331*, *339*
- Short Cut Keys, *834*
- syntax check, *382*
- Tag, *334*

IL command
- call function block, *372*

IL Command
- Comments, *338*

IL command
- Compare, *359*, *360*, *363*

IL Command
- Compare, *361*, *362*, *364*
- Declaration, *336*

IL command
- DFB invocation, *372*
- invert, *352*
- Reset, *344*
- Set, *342*
- VAR...END_VAR, *336*

IL operation
- addition, *353*
- Boolean AND, *346*
- Boolean exclusive OR, *350*
- Boolean OR, *348*
- call DFB, *368*
- call function block, *368*
- jump to label, *365*
- Load, *340*
- multiplication, *355*
- Store, *341*
- subtraction, *354*

IL-Befehl
- Function call, *379*

IL-operation
- division, *357*

Import, *691*
- PLC Configuration, *728*
- General Information, *692*
- Multiple Address Assignment, *725*

import
- PLC configuration, *727*
- section, *699*, *709*

Import
- Section, *698*, *703*, *712*, *713*, *714*
- Structured variables, *720*

import
- variables, *717*

Import
- Variables, *716*, *720*, *724*

import INTERBUS configuration , *962*

importing Profibus DP configuration , *934*

INC
- Include File, *572*

Include File
- Extended Data Type Definition, *572*

INI File
- Event Viewer Settings, *1122*

INI file
- general, *1107*

INI File
- General Information, *1121*

INI file
- LD section settings, *1114*
- print settings, *1108*

INI File
- Project Name Definition, *1109*
- Project Specific, *1105*
- Register Address Format Settings, *1109*

INI file
- representation of internal data, *1113*

INI File
- Security Settings, *1117*
- Setting for Online Processing, *1115*
- Settings for the Address Format, *1116*

INI file
- Settings for the Tools menu, *1119*

INI File
  Settings for Warning Messages, *1116*
  Variable Storage Settings, *1109*
INI Files, *1105*
  CONCEPT.INI, *1106*
  Projectname.INI, *1120*
INI-File
  Exclusion of the global/local DFBs from Online-Backup, *1110*
INI-file
  Path for global DFBs, *1110*
  Path for Help files, *1110*
INI-File
  Reading global DFBs, *1110*
INI-file
  Saving the global DFBs during upload, *1110*
INI-File
  Settings for the Online-Backup, *1123*
  Settings for the RDE, *1118*
initial step, *275*
insert
  FFB, FBD, *225*
  FFB, LD, *258*
Install loadables, *80*
installing the EXEC file , *1101*
installing the Modbus Plus driver
  Windows 98/2000/NT, *1018*
Installing the SA85/PC185
  Modbus Plus Preferences, *1016*
  Windows NT, *1016*
Installing the SA85/PCI85
  Modbus Plus Preferences, *1012*
  Windows 98/2000/XP, *1012*
Instruction
  IL, *324*, *325*
Instruction list, *321*
Instructions
  ST, *423*
INTERBUS controller, *911*
INTERBUS export settings in CMD, *957*
Interface Settings in Windows 98/2000/XP
  Modbus Preferences, *1026*
Interface settings in Windows NT
  Modbus Presettings, *1028*
Interrupt Processing, *1125*
  General, *1126*
Interrupt Sections
  Disable, *70*
  EFBs, *1146*
  Examples for Setting Parameters, *1137*
  Execution Order, *1134*
  I/O Event Sections, *1142*
  Operating System, *1135*
  Priority, *1143*
  Runtime Error, *1144*
  Scan Rate for Timer Event Sections, *1130*
  Timer Event Sections, *1129*, *1131*
invocation
  DFB, *372*, *442*
  FFB, *372*, *442*, *446*
Invoke
  Project, *811*

J

Jump
  SFC, *285*

K

Key combinations, *819*, *832*, *833*, *834*, *837*, *841*, *847*
key words
  data type editor, *577*
  derived data type, *577*
Keys, *819*, *832*, *833*, *834*, *837*, *841*, *847*

L

Ladder Diagram, *239*
Ladder Logic 984, *455*

LD, *239*
- actual parameters, *254*
- animation, *265*
- Calling a macro, *540*
- Closer, *243*
- code generation, *264*
- Coil - negated, *246*
- Coil - reset, *247*
- Coil - set, *247*
- Coil – negative edge, *246*
- Coil – positive edge, *246*
- Coils, *245*
- Contacts, *243*, *243*, *244*
- Data flow, *260*
- derived function block, *250*
- EFB, *248*
- elementary function, *248*
- elementary function block, *249*
- EN, *252*
- ENO, *252*
- Execution sequence, *260*
- FFB, *248*
- function, *248*
- function block, *249*
- Icon bar, *825*
- link, *253*
- loops, *262*
- online functions, *265*
- Opener, *243*
- program creation, *267*
- Shortcut keys, *841*
- Text object, *256*
- UDEFB, *251*
- user-defined elementary function, *251*
- user-defined elementary function block, *251*

Learn monitoring times
- SFC, *316*

Libraries, *31*

Limitations
- LL984, *457*, *457*

link
- FBD, *220*
- LD, *253*

List of symbols, *819*

List of Symbols, *827*, *828*

List of tools, *819*

List of Tools, *827*, *828*

Literals, *64*

LL984, *455*
- Close Column, *465*
- Combination mode, *480*
- Dialog interaction, *459*
- Direct programming, *480*
- DX Zoom, *466*
- Edit, *459*, *464*
- Editing Networks, *465*
- Equation network, *471*, *472*
- Equation network, Syntax and Semantics, *476*
- List of Symbols, *828*
- Momentum first startup, *1044*, *1059*, *1080*, *1098*
- Navigation, *459*
- Online Restriction, *460*
- Online Search, *468*
- Open Column, *465*
- Open Row, *465*
- Programming modes, *480*, *480*
- Reference Offset, *463*
- Reference Zoom, *466*
- References, *461*
- Replace References, *468*
- Requirements, *459*
- Section, *456*
- Segement, *456*
- Select, *464*
- Short Cut Keys, *847*
- Subroutines, *469*
- Trace, *468*
- Undo, *464*
- Variables, *461*

LL984 Processing
- speed optimized, *655*

LL984 section
- Animation, *684*

Load reference data, *609*

loadables, *114*
- Atrium, *207*
- compact, *190*

Loadables
CPU 424 02, *171*
loadables
CPU 434 12, *179*
CPU 534 14, *179*
Loadables
CPU X13 0X, *171*
Loading, *672*
Loading a project, *670*, *670*
General information, *671*
loading firmware, *1101*
local derived data type, *570*
local DFB, *485*
local macro, *525*
Located variables
Changing signal states in RDE, *600*
Log Encoding, *39*
LOG File, *687*, *687*
Logging
LOG File, *687*
Logging Write Access to the PLC, *687*
loop
FBD, *230*
loops
LD, *262*

M

Macro, *521*, *523*
Calling up from SFC, *537*
Calls from FBD, *540*
Calls from LD, *540*
macro
context sensitive help, *529*
Macro
Create, *531*
Delete, *743*
Documentation, *730*
Exchange marking, *527*
macro
global, *525*
local, *525*
Macros
Convert, *991*
maximum supervision time, *275*
MBX Driver
Driver for connection between ModConnect Host interface adapters and 32 bit applications with Windows 98/2000/NT, *1020*
Memory, *155*
Optimize, *158*
PLC-Independent Memory Optimization, *159*
Structure, *156*
Memory and optimization
Atrium, *204*
Compact, *187*
Momentum, *197*
Quantum, *162*, *176*
Memory partitions, *79*
Memory statistics, *667*
Menu commands, *806*
minimum configuration, *79*
minimum supervision time, *276*
MMS-Ethernet
specify coupling modules, *124*
Modbus
Compact first startup, *1036*, *1053*
Momentum first startup, *1040*, *1044*, *1056*, *1059*
Quantum first startup, *1032*, *1050*
Startup with DOS Loader, *1049*
Startup with the EXECLoader, *1031*
Modbus communication, *81*
Modbus network link, *638*
Modbus Plus
Atrium first startup, *1072*, *1092*
Compact first startup, *1068*, *1089*
Momentum first startup, *1076*, *1080*, *1095*, *1098*
Quantum first startup, *1064*, *1086*
Remote MBX Driver, *1021*
Startup with DOS Loader, *1085*
Startup with the EXECLoader, *1063*
Virtual MBX Driver, *1019*
Write Access Restrictions, *151*
Modbus Plus Bridge, *644*
Modbus Plus Network Connection, *639*
Modbus Plus network node, *125*

Modbus Plus preferences
- installing the Modbus Plus driver in Windows 98/2000/NT, *1018*

Modbus Plus Preferences
- Installing the SA85/PC185, *1016*
- Installing the SA85/PCI85, *1012*

Modbus Plus presettings
- Establishing the hardware connection, *1024*
- Startup, *1011*

Modbus Plus Routing Path
- Automatic Connection, *1148*, *1151*

Modbus Preferences
- Interface Settings in Windows 98/2000/XP, *1026*

Modbus presettings
- Establishing the hardware connection, *1029*
- Startup, *1025*

Modbus Presettings
- Transfer problems, *1030*

ModConnect, *995*

MODIFIED, *634*

Modifier
- IL, *329*

Modsoft
- Convertion, *1001*
- Function compatibility, *1010*
- References, *1007*

Momentum
- Memory optimization, *197*

Momentum example
- Ethernet bus system, *974*
- Remote I/O bus, *965*

Momentum first startup
- DOS Loader, *1056*, *1059*, *1095*, *1098*
- EXECLoader, *1040*, *1044*, *1076*, *1080*
- Modbus, *1040*, *1044*, *1056*, *1059*
- Modbus Plus, *1076*, *1080*, *1095*, *1098*

MSTR-Read-Operation, *152*

N

Names
- Datatype editor, *582*
- Derived datatype, *582*

Navigation
- LL984, *459*

Network Configuration
- TCP/IP, *976*

Network Connection
- Modbus Plus, *639*

network link
- Modbus, *638*
- TCP/IP, *646*

NOM/NOE
- Disabling Write Access, *151*

NOT EQUAL, *634*

O

objects
- insert, LD, *258*

Objects
- SFC, *274*

offline functions in the configurator, *105*

Online, *746*, *748*
- INI File, *1115*
- SFC, *310*

Online Control Panel, *656*, *660*

online control panel
- general information, *650*

online diagnosis, *685*

online diagnostics, *685*

Online functionen
- IL, *389*

Online functions, *37*

Online Functions
- FBD, *233*

Online functions
- General information, *632*, *632*
- IL, *385*
- IL/ST, *386*

online functions
- LD, *265*
- SFC, *311*, *313*

Online functions
ST, *451*
online functions in the configurator, *105*
Online help, *813*
ONLINE Operation
Presettings, *637*
Online Restriction
LL984, *460*
Online Search
LL984, *468*
Online-Backup
INI-Settings , *1123*
Online-Funktionen, *631*
Open
Project, *811*, *811*
Open Column
LL984, *465*
Open Row
LL984, *465*
Opener
LD, *243*
Operands
IL, *327*
ST, *397*
Operating System
Timer Event Sections, *1135*
Operators
IL, *331*, *339*
operators
ST, *399*
Operators
ST, *402*
Optimize
PLC Memory, *158*
PLC-Independent Memory Optimization, *159*
Optional Configuration, *122*

P

page breaks for sections, *735*
Parallel branch, *290*
Parallel connection, *291*
Parameterize ASCII interface, *129*
Parameterize interfaces
ASCII interface, *129*
Modbus interface, *129*
Parameterize Modbus interface, *129*
Parameters for Automatic Connection, *811*
Password protection, *757*
Password Protection, *758*, *760*, *767*, *769*
Path for global DFBs
Settings in the INI-File, *1110*
Path for Help files
Settings in the INI-File, *1110*
Peer Cop, *125*, *942*
Peer Cop communication, *82*
performance
hardware, *775*
PLC family, *775*
Phase
Timer Event Sections, *1131*
PLC
Simulating, *745*
Status, *803*
PLC configuration, *78*, *79*, *99*
export, *727*
General information, *101*
icons, *829*
import, *727*
PLC Connection
General, *634*
PLC family
performance, *775*
PLC Hardware Package Contents in Concept S, M and XL, *28*
PLC Memory, *155*
Optimize, *158*
PLC-Independent Memory Optimization, *159*
Structure, *156*
PLC Memory and optimization
Atrium, *204*
Compact, *187*
Momentum, *197*
PLC memory and optimization
Quantum, *176*
PLC Memory and optimization
Quantum, *162*

PLC memory mapping, *113*
PLC selection, *109*
PLC state, *665*
PLC State, *648*
PLC Status, *634*
position
  FFB, FBD, *225*
  FFB, LD, *258*
precondition for unconditional configuration, *108*
Presettings for Modbus
  Startup, *1025*
Presettings for Modbus Plus
  Startup, *1011*
Presettings for ONLINE operation, *637*
print
  settings in the INI file, *1108*
printing sections, *735*
Priority
  I/O Event Sections, *1143*
proceed in the following way with the configuration, *102*
Process
  Actions, *299*
  Step properties, *297*
process
  transition, *305*
Processing
  Program, *58*
  Project, *58*
PROFIBUS
  specify coupling modules, *124*
Profibus DP export settings in SyCon, *927*
Program
  Create, *75*
  Processing, *58*
  Status, *803*
  Structure, *57*, *58*
Program creating
  ST, *452*
program creation
  FBD, *235*
Programming, *29*
Programming languages, *33*
Programming modes
  LL984, *480*, *480*
Programs, *64*
Project
  Archiving, *741*
  Call, *811*
  Create, *75*
  Delete, *743*
  Documentation, *730*
  Invoke, *811*
  Open, *811*, *811*
  Processing, *58*
  Protect, *770*
  Structure, *57*, *58*
Project Browser, *557*
  Keyboard operation, *563*
  Mouse operation, *563*
  Toolbar, *831*
Project Name Definition
  INI File Settings, *1109*
Project Symbol
  Create, *811*
Project symbol
  Generate, *811*
Projectname.INI, *1105*, *1120*
  Event Viewer Settings, *1122*
  General Information, *1121*
Projects
  Convert, *991*
Projektname.INI
  Settings for the Online-Backup, *1123*
Protect
  DFB, *770*
  Project, *770*

Q

Quantum
  Memory optimization, *162*, *176*

Quantum example
INTERBUS control, *910*
Profibus DP controller, *925*
Quantum-Peer Cop, *941*
Remote control with DIO, *900*
Remote control with RIO, *878*
Remote control with RIO (series 800), *887*
SY/MAX controller, *916*
Quantum first startup
DOS Loader, *1050*, *1086*
EXECLoader, *1032*, *1064*
Modbus, *1032*, *1050*
Modbus Plus, *1064*, *1086*
Quantum Security Parameters, *151*

R

Range Monitoring
ARRAY, *590*
RDE, *595*
converting RDE templates, *598*
Cyclical Setting of Variables, *602*
General, *596*
Settings in the INI-File, *1118*
RDE editor
toolbar, *830*
reactivate flash save, *659*
Reading the global DFBs
Settings in the INI-File, *1110*
Reference data editor, *595*
Changing signal states of a Located variable, *600*
reference data editor
converting RDE templates, *598*
Reference Data Editor
Cyclical Setting of Variables, *602*
General, *596*
Reference data editor
Replacing variable names, *608*
Reference Offset
LL984, *463*
Reference Zoom
LL984, *466*
References
LL984, *461*
Register Address Format
INI File Settings, *1109*
Remote controller with DIO, *906*
Remote controller with RIO, *884*
Remote controller with RIO (series 800), *893*
Remote MBX Driver
Modbus Plus, *1021*
replace
coil, LD, *259*
contact, LD, *259*
FFB, FBD, *226*
FFB, LD, *259*
Replace References
LL984, *468*
Replacing variable names
Reference data editor, *608*
Requirements
LL984, *459*
RTU extension
Compact configuration, *143*
Configure, *143*
Runtime Error
I/O Event Sections, *1144*

S

Save To Flash, *656*
Saving the global DFBs during Upload
Settings in the INI-File, *1110*
Scan
Constant, *651*
Scan Rate
Timer Event Sections, *1130*
scan times
single, *652*
Search and Replace
Variable names and addresses, *548*
searching and pasting
variable names and addresses, *552*
searching and pasting variable names and addresses, *552*

Section, *68*
- Animation, *681*
- Disable, *69*
- Execution order, *69*
- Export, *694*

section
- import, *709*

Section
- import, *699*
- Import, *698*, *703*, *712*, *713*, *714*
- LL984, *456*
- Status, *803*

Secure Application, *39*

Security, *757*, *758*, *760*, *767*, *769*, *770*

Segement
- LL984, *456*

Segment manager, *117*

Select
- LL984, *464*

Select process information
- General information, *664*
- Status and memory, *663*

Selecting process information
- Status and memory, *663*

separators
- data type editor, *583*
- derived data type, *583*

Set/Change PLC Password, *660*

Setting up and controlling the PLC, *649*, *649*, *649*

setup and control PLC
- general information, *650*

SFC
- ’SFCSTEP_STATE’ variable, *277*, *277*
- ’SFCSTEP_TIMES’ variable, *276*
- Action, *278*
- Action variable, *278*
- Actions, *299*
- alternative branch, *287*
- Alternative connection, *289*
- animation, *311*, *313*
- Calling up macros, *537*
- edit, *293*
- Edit, *294*
- Icon bar, *823*
- Identifier, *302*
- initial step, *275*
- Jump, *285*
- Learn monitoring times, *316*
- Link, *284*
- maximum supervision time, *275*
- minimum supervision time, *276*
- Objects, *274*
- Online, *310*
- online functions, *311*, *313*
- Parallel branch, *290*
- Parallel connection, *291*
- Short Cut Keys, *837*
- Step, *275*
- step delay time, *275*
- step duration, *275*
- Step properties, *297*
- string, *313*
- Text object, *292*
- transition, *280*, *280*, *305*
- Transition diagnosis, *319*
- Transition section, *282*
- Transition variable, *283*
- waiting step, *275*

Short cut keys, *819*, *832*

Short Cut Keys, *833*, *834*, *837*, *847*

Shortcut keys, *841*

Simple sequences, *284*

Simulate
- SPS, *746*, *748*

Simulation, *745*, *746*, *748*

single sweeps , *652*

special options, *131*
Specific data transfer
  Peer Cop, *948*
Speed optimized LL984- Processing, *655*
SPS
  Simulate, *746*, *748*
ST, *393*
  Animation, *386*
  Assign instructions, *422*
  Block call up, *441*
  Code generation, *450*
  Expressions, *396*
  Instructions, *423*
  List of Symbols, *827*
  Online functions, *386*, *451*
  Operands, *397*
  operators, *399*
  Operators, *402*
  Program creation, *452*
  Short Cut Keys, *834*
  syntax check, *449*
ST command
  -, *406*, *412*
  (), *403*
  *, *408*
  **, *405*
ST Command
  &, *419*
ST command
  +, *411*
ST Command
  <, *417*
  <=, *418*
  <>, *416*
  =, *415*
ST command
  >, *413*
  >=, *414*
  Addition, *411*
ST Command
  AND, *419*
ST command
  Assignment, *424*
ST Command
  Boolean AND, *419*
  Boolean OR, *420*
ST command
  Call function block, *442*
  CASE...OF...END_CASE, *431*
  Complement formation, *407*
  Declaration, *426*
  Division, *409*
ST Command
  ELSE, *429*
  ELSIF...THEN, *430*
ST command
  Empty instruction, *439*
ST Command
  Equal to, *415*
ST command
  EXIT, *438*
  Exponentiation, *405*
  FOR...TO...BY...DO...END_FOR, *432*
  FUNCNAME, *404*
  function invocation, *446*
  Greater than, *413*
  Greater than/Equal to, *414*
  IF...THEN...END_IF, *428*
ST Command
  Less than, *417*
  Less than or equal to, *418*
ST command
  MOD, *410*
  Modulo, *410*
  Multiplication, *408*
  Negation, *406*
  NOT, *407*
ST Command
  Not equal to, *416*
  OR, *420*
ST command
  REPEAT...UNTIL...END_REPEAT, *437*
  Subtraction, *412*
  Use of parentheses, *403*
  VAR...END_VAR, *426*
  WHILE...DO...END_WHILE, *435*
ST Command
  XOR, *421*

ST commandl
    /, *409*
ST Comment
    Comment, *440*
ST-Command
    Boolean Exclusive OR, *421*
Start behavior
    Variable, *65*
Start behavior of digital outputs, *67*
Startup
    Presettings for Modbus, *1025*
    Presettings for Modbus Plus, *1011*
Startup with DOS Loader
    Modbus, *1049*
    Modbus Plus, *1085*
Startup with the EXECLoader
    Modbus, *1031*
    Modbus Plus, *1063*
State of the PLC, *648*
Status, *634*
Status bar, *803*
step, *275*
    alias designations, *307*
step delay time, *275*
step duration, *275*
Step properties
    Process, *297*
string
    control, *313*
Structure
    PLC Memory, *156*
    Program, *57*, *58*
    Project, *57*, *58*
Structured text, *393*
Structured variables
    Import, *720*
Subroutines
    LL984, *469*
Symax-Ethernet
    specify coupling modules, *124*
Symbols, *819*, *827*, *828*
Syntax
    Data type editor, *574*
    Derived Data Type, *574*
syntax check
    IL, *382*
    ST, *449*

T

Tag
    IL, *334*
TCP/IP
    Network Configuration, *976*
TCP/IP network link, *646*
TCP/IP-Ethernet
    specify coupling modules, *124*
Text Object
    FBD, *223*
Text object
    LD, *256*
    SFC, *292*
Timer Event Section
    Handling, *1125*
Timer Event Sections, *1129*
    Define Scan Rate, *1130*
    Defining the Phase, *1131*
    Examples for Parameterization, *1137*
    Execution Order, *1134*
    Operating System, *1135*
Tool bar, *820*, *821*, *822*, *823*, *825*
toolbar, *830*, *830*
Toolbar, *831*, *831*
Tools, *41*
    Settings in the INI file, *1119*
Trace
    LL984, *468*
Transfer problems
    Modbus Presettings, *1030*
transition, *280*, *280*
    alias designations, *307*
    declare, *305*
    process, *305*
Transition diagnosis, *319*
Transition section, *282*
Transition variable, *283*

U

UDEFB
- FBD, *219*
- LD, *251*

Unconditional Configuration, *107*

unconditional configuration
- precondition, *108*

Unconditional locking of a section, *605*

Undo
- LL984, *464*

Upload PLC , *677*

Uploading the PLC, *677*

User-defined Elementary Function
- FBD, *219*

user-defined elementary function
- LD, *251*

User-defined Elementary Function Block
- FBD, *219*

user-defined elementary function Block
- LD, *251*

Utility program, *41*

V

Variable
- Export, *697*
- Start behavior, *65*

Variable Editor
- Declaration, *545*

Variable editor
- Exporting located variables, *556*
- Search and Replace, *548*

variable editor
- searching and pasting, *552*

Variable Storage
- INI File Settings, *1109*

Variables, *64*
- ASCII message editor, *615*

variables
- import, *717*

Variables
- Import, *716*, *720*, *724*
- LL984, *461*

Variables editor, *543*

Variables-Editor
- General, *544*

VARINOUT variables, *489*

Various PLC settings, *83*

View Tool, *687*

Virtual MBX Driver
- Modbus Plus, *1019*

W

waiting step, *275*

Warm restart, *65*

Window, *800*

Window elements, *803*

window types, *801*

Windows, *799*
- Check box, *810*
- Command buttons, *809*
- Dialog boxes, *808*
- Lists, *809*, *809*
- Menu commands, *806*
- Option buttons, *809*
- Status bar, *803*
- Text boxes, *809*
- Window, *800*
- Window elements, *803*

windows
- window types, *801*