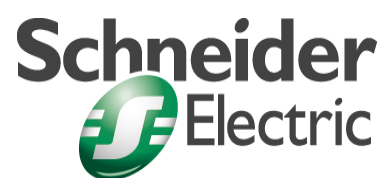


Concept User Manual

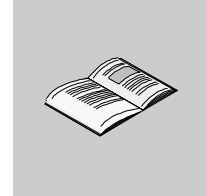
Volume 2

840 USE 503 00 eng Version 2.6 - SR1



© 2002 Schneider Electric All Rights Reserved

Table of Contents



About the Book..... XI

The chapters marked gray are not included in this volume.

Chapter 1	General description of Concept	1
Chapter 2	New Performance Attributes of Concept 2.6 in Comparison with Concept 2.5.....	21
Chapter 3	Project structure.....	29
Chapter 4	Creating a Project.....	47
Chapter 5	PLC configuration.....	69
Chapter 6	Main structure of PLC Memory and optimization of memory	115
Chapter 7	Function Block language FBD.....	173
Chapter 8	Ladder Diagram LD.....	199
Chapter 9	Sequence language SFC	233
Chapter 10	Instruction list IL.....	279
Chapter 11	Structured text ST.....	341
Chapter 12	Ladder Logic 984	387
	At a Glance	387
12.1	General about Ladder Logic 984.....	389
12.2	Working with Ladder Logic 984.....	392
	At a Glance	392
	Entering and Editing Logic Objects.....	393
	Entering and Editing Variables	395

	Ladder and Network Editing.	397
	Reference Zoom and DX Zoom.	399
	Search and Replace	401
12.3	Subroutines	402
12.4	Equation Network Editor	404
	At a Glance	404
	Introduction	405
	Equation Editing.	407
	Syntax and Semantics.	409
12.5	LL984 Programming Modes.	413
Chapter 13	DFBs (Derived Function Blocks)	415
	At a Glance	415
13.1	DFBs (Derived Function Blocks)	417
	At a Glance	417
	General information about DFBs (Derived Function Blocks)	418
	Global / Local DFBs.	420
	Use of variables in DFBs.	422
	Combined Input/Output Variables (VARINOUT Variables)	423
	Global Variables.	430
	Creating Context Sensitive Help (Online Help) for DFBs.	434
13.2	Programming and calling up a DFB.	436
	At a Glance	436
	At a Glance	437
	Creating the DFB.	438
	Creating the Logic in FBD Function Block Language	439
	Creating the Logic in LD Ladder Diagram	441
	Creating the Logic in IL Instruction List	445
	Creating the Logic in ST Structured Text.	447
	Calling up a DFB in the FBD Function Block dialog.	449
	Calling up a DFB in Ladder Diagram LD	451
	Calling up a DFB in the IL instruction list.	453
	Calling up a DFB in structured text ST.	454
Chapter 14	Macros	455
	At a Glance	455
14.1	Macro.	457
	At a Glance	457
	Macros: general.	458
	Global / Local Macros	460
	Exchange marking.	462
	Creating Context Sensitive Help (Online Help) for Macros	465
14.2	Programming and calling up a macro	467
	At a Glance	467
	At a Glance	468
	Occupying the macro.	469

	Creating the logic	470
	Calling up a macro from an SFC section.	473
	Calling a macro from an FBD/LD section.....	476
Chapter 15	Variables editor.....	479
	At a Glance	479
	General	480
	Declare variables.....	480
	Searching and replacing variable names and addresses	483
	Searching and Pasting Variable Names and Addresses.....	487
	Exporting located variables.....	490
Chapter 16	Project Browser	491
	At a Glance	491
	General information about the Project Browser	492
	Detailed view in the project browser	494
	Operating the Project Browser	496
Chapter 17	Derived data types	499
	At a Glance	499
17.1	General information on Derived Data Types.....	501
	At a Glance	501
	Derived Data Types.....	502
	Global / Local Derived Data Types	505
	Extended Data Type Definition (larger than 64 Kbytes)	507
17.2	Syntax of the data type editor	509
	At a Glance	509
	Elements of the Derived Data Types.....	510
	Key Words.....	512
	Names of the derived datatypes	516
	Separators.....	517
	Comments.....	518
17.3	Derived data types using memory.....	520
	Use of Memory by Derived Data Types	521
17.4	Calling derived data types.....	523
Chapter 18	Reference data editor.....	531
	At a Glance	531
	General Information about the Reference Data Editor	532
	Converting RDE templates	533
	Changing signal states of a Located variable	535
	Cyclical Setting of Variables	536
	Unconditional locking of a section.....	538
	Animation	539
	Replacing variable names.....	541
	Load reference data	542

Chapter 19	ASCII Message Editor	543
	At a glance	543
19.1	ASCII Editor Dialog	545
	At a glance	545
	Generals to ASCII editor dialog	546
	Text	547
	Variables	548
	Control code	549
	Spaces	549
	Carriage Return	550
	Flush (buffer)	551
	Repeat	552
19.2	User Interface of ASCII Message Editor	553
	At a glance	553
	How to Use the ASCII Message Editor	554
	Message Number	555
	Message Text	556
	Simulation Text	556
19.3	How to Continue after Getting a Warning	557
	How to Continue after Getting a Warning	558
19.4	ASCII Editor in Offline/Combination/Direct Modes	559
	ASCII Message Editor in Offline/Combination/Direct Modes	560
Chapter 20	Online functions	561
	At a Glance	561
20.1	General information about online functions	563
20.2	Connect to PLC	565
	At a Glance	565
	General	566
	Presettings for ONLINE operation	569
	Modbus Network Link	570
	Modbus Plus Network	571
	Modbus Plus Bridge	576
	TCP/IP-Network Link	578
	Connecting IEC Simulator (32 bit)	578
	State of the PLC	579
20.3	Setting up and controlling the PLC	580
	General Information	581
	Setting the Time for Constant Scans	582
	Single Sweeps	583
	Deleting memory zones from the PLC	584
	Speed optimized LL984-Processing	585
	Save To Flash	585
	Reactivate flash save	588
	Set PLC Password	589

20.4	Selecting Process information (status and memory)	592
	At a Glance	592
	General information	593
	PLC state	594
	Memory Statistics	595
20.5	Loading a project	597
	At a Glance	597
	General information	598
	Loading	599
	Download Changes	600
	Uploading the PLC	603
	Upload Procedure	604
20.6	Section animation	606
	At a Glance	606
	IEC-Sections animation	607
	LL984 Programming Modes	609
20.7	Online Diagnosis	610
	Diagnostics Viewer	611
20.8	Logging Write Access to the PLC	613
	Logging and Encrypted Logging	614
Chapter 21	Import/Export	619
	At a Glance	619
21.1	General Information about Import/Export	621
21.2	Exporting sections	624
21.3	Exporting variables and derived data types	628
21.4	Section import	630
	At a Glance	630
	Importing Sections	631
	Procedure for importing sections	635
	Importing IL and ST Programs to FBD, SFC, IL or ST Sections (with Conversion)	642
	Importing (insert file) IL and ST programs into IL or ST sections	645
	Procedure for "Copying" an IL section from an existing project into a new project	646
	Procedure for converting FBD sections from an existing project into IL sections of a new project	647
21.5	Variables import	649
	At a Glance	649
	Importing Variables in "Text Delimited" Format	650
	Importing structured variables	653
	Importing variables in Factory Link format	656
21.6	Import/Export of PLC Configuration	657
	Introduction	657
	Import/Export of PLC Configuration using Concept	658
	Import/Export of PLC Configuration using Concept Converter	659

Chapter 22	Documentation and Archiving	661
	At a Glance	661
22.1	Documentation of projects, DFBs and macros	663
	At a Glance	663
	Documentation contents	664
	Documentation Layout	665
	Defining Page Breaks for Sections	667
	Use of keywords	671
22.2	Managing projects, DFBs and macros	673
	At a Glance	673
	Archiving projects, used DFBs, EFBs and data type files	674
	Deleting projects, DFBs and macros	676
Chapter 23	Simulating a PLC	677
	Preview	677
23.1	Simulating a PLC (16-bit simulator)	679
	Simulating a Controller	680
23.2	Simulating a PLC (32-bit simulator)	682
	At a Glance	682
	Concept-PLCSIM32	683
	Simulating a PLC	685
	Simulating a TCP/IP interface card in Windows 98	686
	Simulating a TCP/IP interface card in Windows NT	687
Chapter 24	Concept Security	691
	At a Glance	691
	General Description of Concept Security	692
	Access Rights	694
	Changing Passwords	701
	Activating Access Rights	702
	Protecting Projects/DFBs	702
Appendices		705
	At a Glance	705
Appendix A	Tables of PLC-dependent Performance Attributes	707
	Introduction	707
	Performance of Quantum	708
	Performance Attributes of Compact	713
	Performance Attributes of Momentum	717
	Performance Attributes of Atrium	722

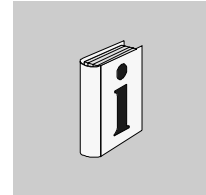
Appendix B	Windows interface	729
	At a Glance	729
B.1	Window	731
	At a Glance	731
	Window Types	732
	Elements of a window	733
B.2	Menu commands	736
B.3	Dialog boxes	739
B.4	Generating a project symbol	743
	Creating a Project Symbol in a Program Group	744
B.5	Online help	746
	At a Glance	747
	How the Online Help is set out	748
Appendix C	List of symbols and short cut keys	751
	At a Glance	751
C.1	Icon bar	753
	At a Glance	753
	General icon bar	754
	Icon bar in the FBD editor	755
	Icon bar in the SFC-Editor	756
	Icon bar in the LD editor	758
	List of Symbols in the IL and ST Editor	759
	List of Symbols in the LL984-Editor	760
	Icons in PLC Configuration	761
	Toolbar in the RDE Editor	762
	Toolbar in the Project Browser	762
C.2	Short cut keys	763
	At a Glance	763
	General Short Cut Keys	764
	Short Cut Keys in the IL, ST and Data Type Editor	765
	Short Cut Keys in the FBD and SFC Editor	768
	Shortcut keys in the LD-Editor	771
	Short Cut Keys in the LL984-Editor	777
Index		i

The chapters marked gray are not included in this volume.

Appendix D	IEC conformity	779
Appendix E	Configuration examples	805

Appendix F	Convert Projects/DFBs/Macros	911
Appendix G	Concept ModConnect	915
Appendix H	Conversion of Modsoft Programs.....	923
Appendix I	Modsoft and 984 References	929
Appendix J	Presettings when using Modbus Plus for startup	933
Appendix K	Presettings when using Modbus for startup.....	947
Appendix L	Startup when using Modbus with the EXECLoader	953
Appendix M	Startup when using Modbus with DOS Loader.....	973
Appendix N	Startup when using Modbus Plus with the EXECLoader...	987
Appendix O	Startup when using Modbus Plus with DOS Loader	1007
Appendix P	EXEC files.....	1023
Appendix Q	INI Files	1027
Appendix R	Interrupt Processing	1041
Appendix S	Automatic Connection to the PLC	1067
Glossary	1077

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

User Comments We welcome your comments about this document. You can reach us by e-mail at TECHCOMM@modicon.com

Ladder Logic 984

12

At a Glance

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	389
12.2	Working with Ladder Logic 984	392
12.3	Subroutines	402
12.4	Equation Network Editor	404
12.5	LL984 Programming Modes	413

12.1 General about Ladder Logic 984

General about Ladder Logic 984

Introduction	<p>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.</p>
Correlation between Sections and Segments	<p>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.</p>
Using the Keyboard	<p>Editing in Concept is ordinarily done with the mouse, but it is also possible with the keyboard (see also <i>Short Cut Keys in the LL984-Editor, p. 777</i>).</p>
Project Analyzation	<p>Ladder logic is analyzed before the program is downloaded to the controller.</p> <p>The editor permits only valid Ladder Logic to be entered in the editor, e.g.:</p> <ul style="list-style-type: none">• 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. <div>Note: Your changes to configuration may cause the program to become incompatible with the configuration.</div> <div>Note: Contacts or coils may be entered without references. This not allowed, but not covered by the project analyzation.</div>

**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

At a Glance

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	393
Entering and Editing Variables	395
Ladder and Network Editing	397
Reference Zoom and DX Zoom	399
Search and Replace	401

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).

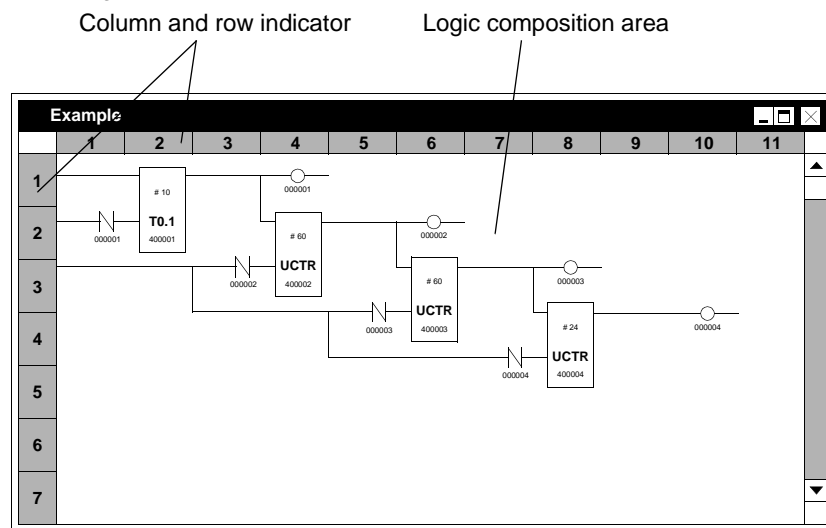
You can go to a network within a section by using the **Go to Network** dialog. You can select the first or last network within the current section, or go to a network by entering a network name or number. A sortable list of networks (with names) is provided.

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, p. 777*). 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, p. 930*).

Online Restriction

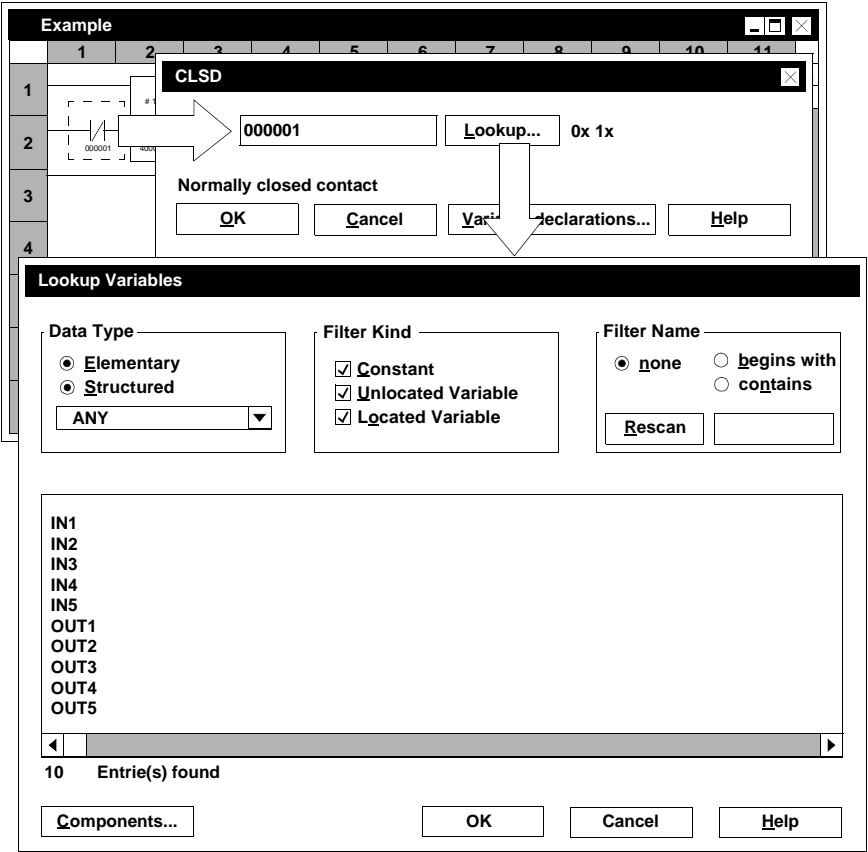
Online restrictions:

- 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:



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 offline 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 thereby 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). You 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	<p>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.</p> <p>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</p>
Reorder Networks	<p>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.</p> <p>The execution order of networks is changed by using the Network Execution Order dialog. i.e. select Network → Reorder...</p>
Network Comments	<p>A section description can be included. Each network can be individually commented using network comments and online comments.</p> <p>A network name can be entered in the Network Comment dialog.</p>

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.

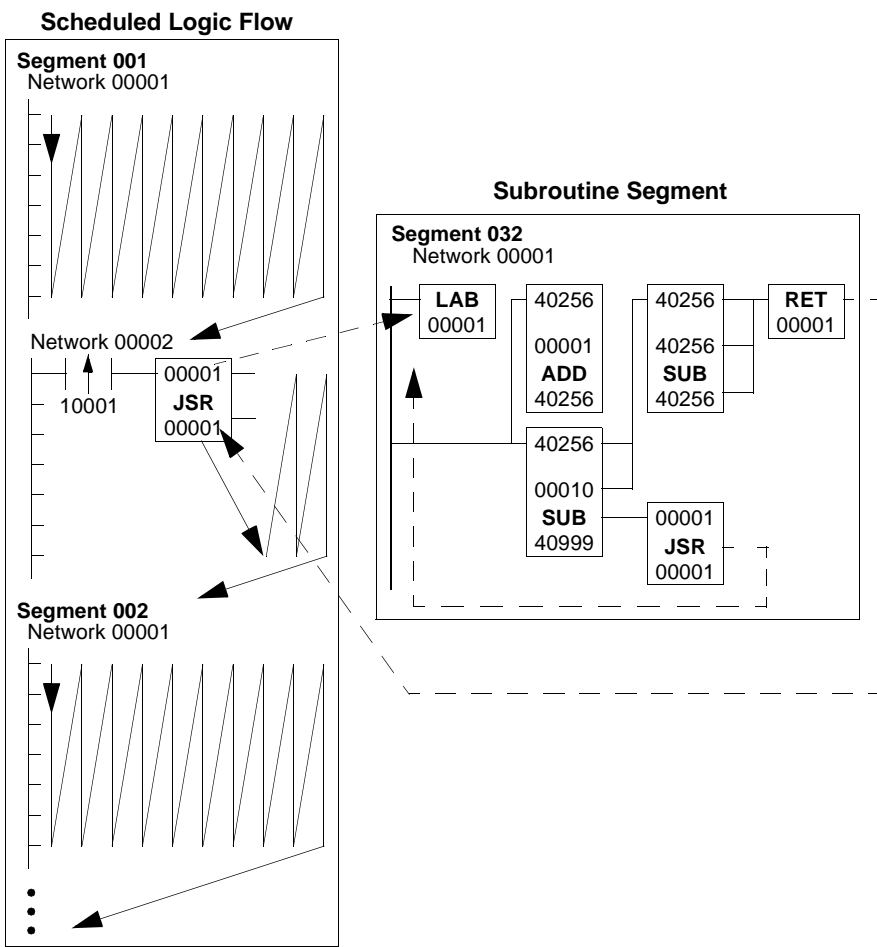
The **Edit** → **Replace References** dialog is modal. Request may be prompted for each individual replace, or request to replace all with no prompting. Replaced references are listed in the **Project** → **Search** → **Search History** list.

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

At a Glance

Introduction This section describes the LL984 equation network editor.

What's in this Section? This section contains the following topics:

Topic	Page
Introduction	405
Equation Editing	407
Syntax and Semantics	409

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 a 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.

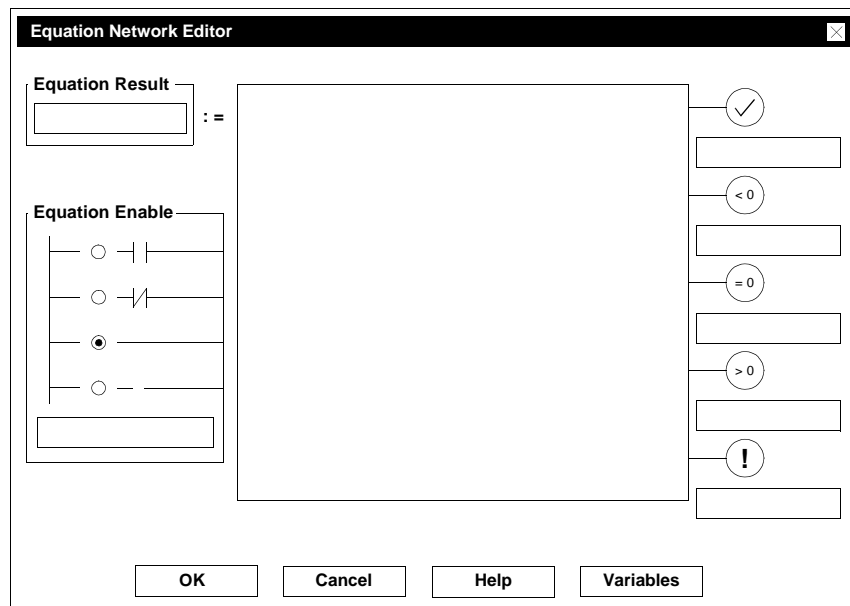
Available Menu Items

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.



The **Equation Network Editor** dialog box is shown. It features a central workspace for the equation network. On the left, there is a section for **Equation Result** with a text input field and a **:=** operator. Below this is the **Equation Enable** section, which contains four radio button options with corresponding ladder logic symbols: a normally open contact, a normally closed contact, a latching contact (selected), and a reset contact. At the bottom of the enable section is a text input field. On the right side of the workspace, there is a vertical stack of five comparison operators in circles: **>**, **<**, **=**, **>**, and **!**. Each operator is followed by a text input field. At the bottom of the dialog are four buttons: **OK**, **Cancel**, **Help**, and **Variables**.

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 ($-(\surd)$)
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 network 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
	?:	Test
Conditional		

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
EXP	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 Loarithm (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 digit+

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 digit+

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 labeled **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 original 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

At a Glance

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)	417
13.2	Programming and calling up a DFB	436

13.1 DFBs (Derived Function Blocks)

At a Glance

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)	418
Global / Local DFBs	420
Use of variables in DFBs	422
Combined Input/Output Variables (VARINOUT Variables)	423
Global Variables	430
Creating Context Sensitive Help (Online Help) for DFBs	434

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
-

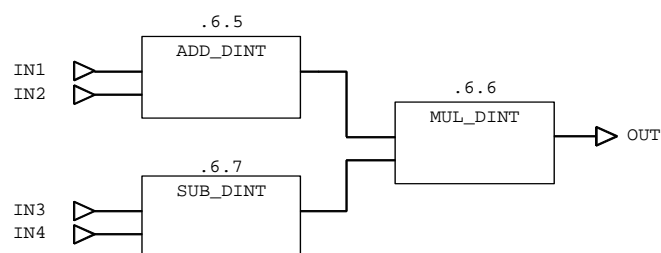
Programming languages

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

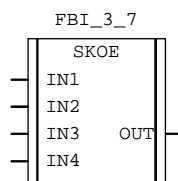
Personalized context-sensitive help (online help) can be created for DFBs (see *Creating Context Sensitive Help (Online Help) for DFBs*, p. 434).

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*, p. 661).

Global / Local DFBs

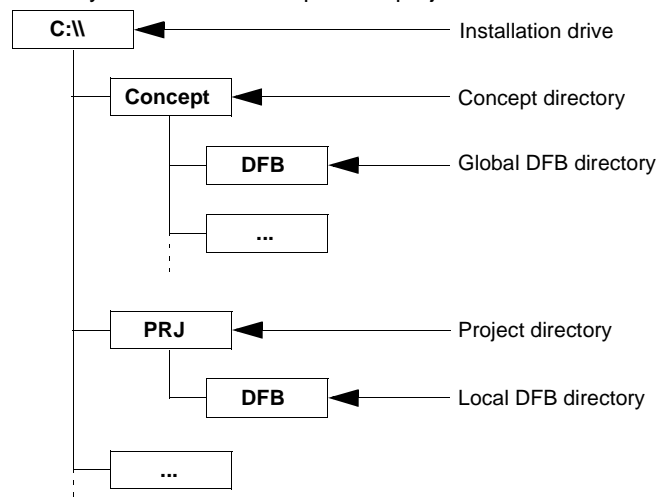
Description

Global and local DFBs differ in the locality of their directory hierarchy.

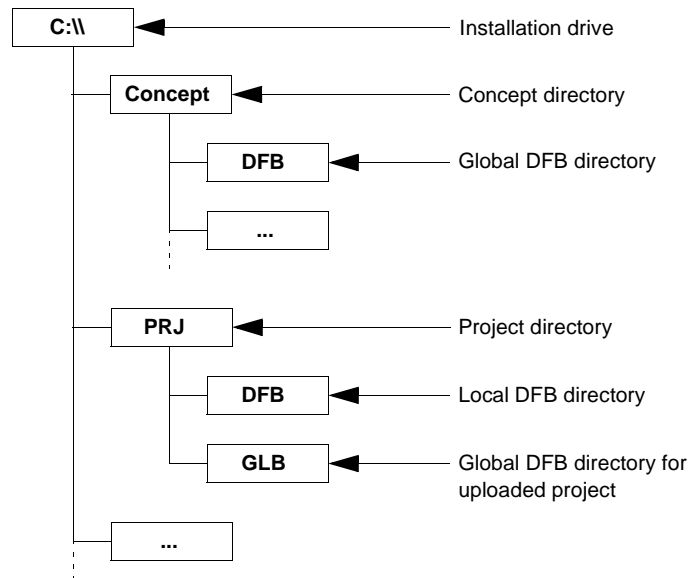
Depending on the directory or subdirectory in which the DFB is stored, it can be called up globally, i.e. within all the projects created under Concept, or locally, in a specific project.

In the *Defining the Storage of Global DFBs during Upload*, p. 1033 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 Extensions** dialog box you activate the **Allow located variables in DFBs** option (see also section *Global Variables*, p. 430).

These variables are declared in the Variable editor (See *Global / Local DFBs*, p. 420).

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 *Global / Local DFBs*, p. 420) 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 *Combined Input/Output Variables (VARINOUT Variables)*, p. 423).

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.

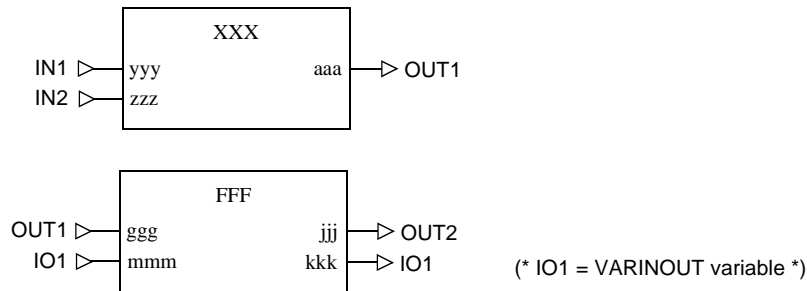
**Specific
Features during
Creation**

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:



Declaration of inputs:

Variable Editor

Type

☐ Variables ☐ Constants ☒ Inputs ☐ Outputs

Search/Paste

Search/Replace

	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						

OK

Cancel

Help

Declaration of outputs:

Variable Editor

Type

☐ Variables ☐ Constants ☐ Inputs ☒ Outputs

Search/Paste

Search/Replace

	Variable Name	Data Type	Position	Used	
1	OUT	REAL	▼ 1	1	
2	OUT2	REAL	▼ 3	1	
3	IO1	MYTYPE	▼ 2	2	
4			▼		
5			▼		
6			▼		

OK

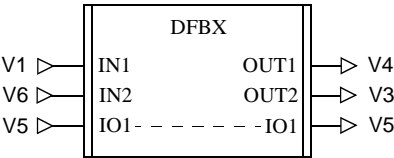
Cancel

Help

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, p. 449* and *Calling up a DFB in Ladder Diagram LD, p. 451*) 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*, p. 373) 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*, p. 321) 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 **Address** column is available in the DFB Variable Editor, i.e. located variables can now be declared.

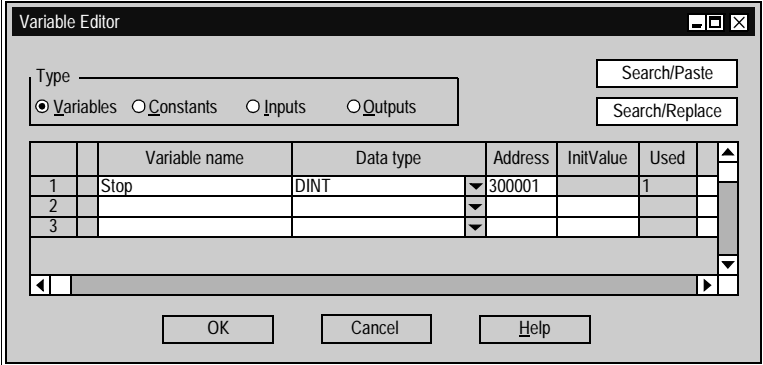
Global validity of the variables is defined as soon as the DFB is used in the project and the respective located variables are declared in the Concept Variable Editor. When declaring the variables, make sure that the same name, address and data type is used as in the DFB Variable Editor. All reference ranges can be used (0x, 1x, 3x and 4x).

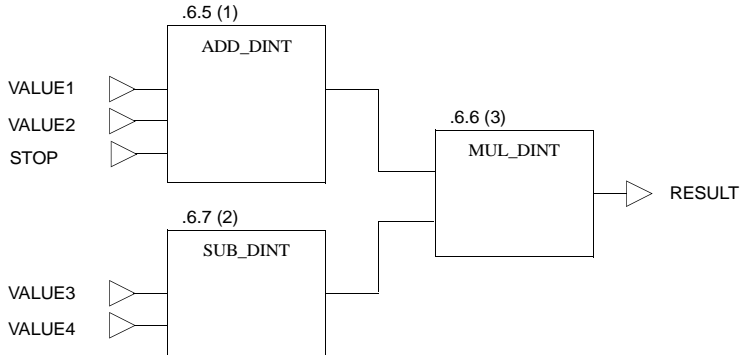
Declaration errors are found and error messages are given when the program is analyzed (**Project** → **Analyze program**). If global validity is recognized, the global variables are shown with a gray background in the Concept Variable Editor and are write protected in Concept. That means global variables can only 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**

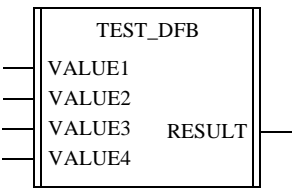
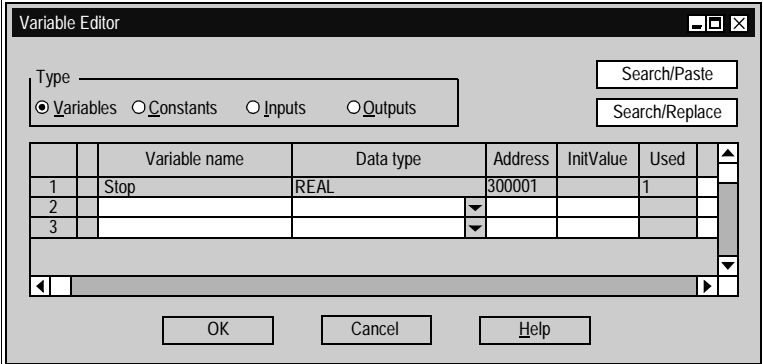
To create global variables in DFB, carry out the following steps in Concept-DFB:

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 <i>Creating the DFB</i> , p. 438).
4	Create the logic (example: see section <i>Creating the Logic in FBD Function Block Language</i> , p. 439).
5	<p>Select Project → Variable declarations. To declare the located variables, activate the Variables option button.</p> <p>Note: All reference ranges can be used (0x, 1x, 3x and 4x) for addressing.</p> 
6	<p>Now re-activate the selection mode with Objects → Select mode and double-click on one of the unconnected inputs.</p> <p>Result: The Connect FFB dialog box is opened, where you can assign a current parameter to the input.</p>
7	In Connect with , activate the Variable option button.
8	<p>Open the variable editor using the Variable declaration... command button. Then select the unlocated variable (STOP) and click OK.</p> <p>Result: The selected variable is transferred to the text box in the Connect with dialog box.</p>

Step	Action
9	<p>With OK, the variable (STOP) is assigned to the selected input on the module.</p>  <pre> graph LR V1[VALUE1] --> A[.6.5 (1) ADD_DINT] V2[VALUE2] --> A S[STOP] --> A A --> M[.6.6 (3) MUL_DINT] V3[VALUE3] --> B[.6.7 (2) SUB_DINT] V4[VALUE4] --> B B --> M M --> R[RESULT] </pre>
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:

Step	Action
1	Close the Concept DFB and Start Concept.
2	<p>Call the DFB (example: see section <i>Calling up a DFB in the FBD Function Block dialog, p. 449</i>).</p> <p>FBI_1_1(1)</p> 
3	Select Project → Variable declarations.... . To declare the located variables (STOP), activate the Variables option button.
4	Transfer the variable names, data type and the address of the located variables, exactly as they were declared in the Concept-DFB variable editor.
5	<p>Analyze the program using Project → Analyze program.</p> <p>Result: The Messages window is opened and shows that the global variable "STOP" was found in the DFB.</p> <p>The global validity of the variable is recognized, therefore it is shown with a gray background in the Concept Variable Editor.</p> 
6	In the DFB Editor, you can open the Function Block dialog box by double-clicking on the DFB. Using the Refine... command button, open a document window with the inner logic of the DFB (the global variable STOP is also shown here).

Creating Context Sensitive Help (Online Help) for DFBs

Introduction	<p>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.</p>
File Format:	<p>You can create your help in the following file formats:</p> <ul style="list-style-type: none">• .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	<p>The name of the help file must be exactly the same as the name of the DFB (e.g. SKOE.ext)</p> <p>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).</p>
Directory	<p>The help file can be stored in the following directories:</p> <ul style="list-style-type: none">• 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	<p>Search for the help file DFBName.ext in the local DFB-directory. The help file is searched for in the following sequence:</p> <ul style="list-style-type: none">• .hlp• .chm• .htm• .rtf• .doc• .txt• .pdf <p>Result: If the search result is positive the help file will be displayed, otherwise it will continue with phase 2.</p>
2	<p>Search for the help file DFBName.ext in the local DFB-directory. For the order, see phase 1.</p> <p>Result: If the search result is positive the help file will be displayed, otherwise it will continue with phase 3.</p>
3	<p>Search for the help file DFBName.ext in the Concept-directory or Concept-Help directory. For the order, see phase 1.</p> <p>Result: If the search result is positive the help file will be displayed, otherwise it will continue with phase 4.</p>
4	<p>Display of the comment created in Concept DFB with Project → Properties.</p>

13.2 Programming and calling up a DFB

At a Glance

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	437
Creating the DFB	438
Creating the Logic in FBD Function Block Language	439
Creating the Logic in LD Ladder Diagram	441
Creating the Logic in IL Instruction List	445
Creating the Logic in ST Structured Text	447
Calling up a DFB in the FBD Function Block dialog	449
Calling up a DFB in Ladder Diagram LD	451
Calling up a DFB in the IL instruction list	453
Calling up a DFB in structured text ST	454

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 <i>Creating the DFB</i> , p. 438)
2	Creating the logic in: <ul style="list-style-type: none">● Function Block language (FBD) (See <i>Creating the Logic in FBD Function Block Language</i>, p. 439)● Ladder diagram (LD) (See <i>Creating the Logic in LD Ladder Diagram</i>, p. 441)● Instruction list (IL) (See <i>Creating the Logic in IL Instruction List</i>, p. 445)● Structured text (ST) (See <i>Creating the Logic in ST Structured Text</i>, p. 447)
3	Calling up the DFB in: <ul style="list-style-type: none">● Function Block language (FBD) (See <i>Calling up a DFB in the FBD Function Block dialog</i>, p. 449)● Ladder diagram (LD) (See <i>Calling up a DFB in Ladder Diagram LD</i>, p. 451)● Instruction list (IL) (See <i>Calling up a DFB in the IL instruction list</i>, p. 453)● Structured text (ST) (See <i>Calling up a DFB in structured text ST</i>, p. 454)

Creating the DFB

Description

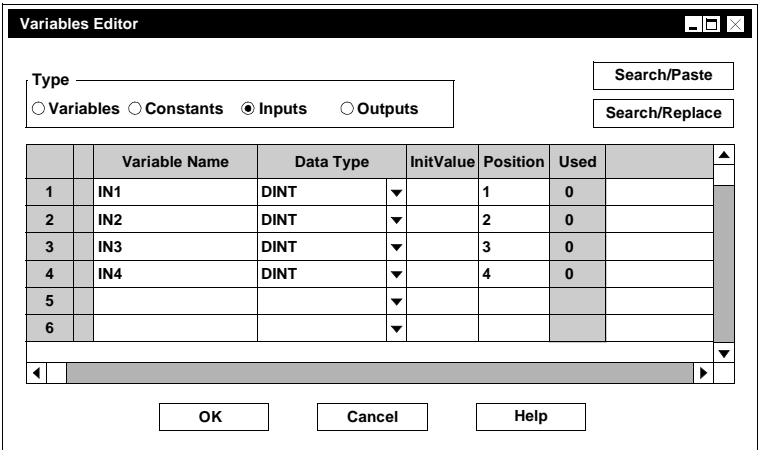
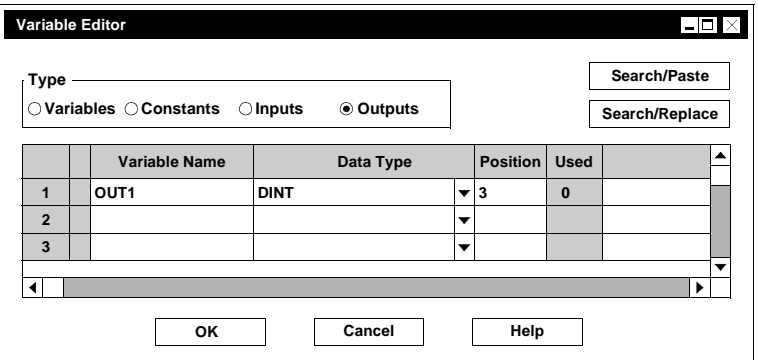
The procedure for creating the DFB is as follows:

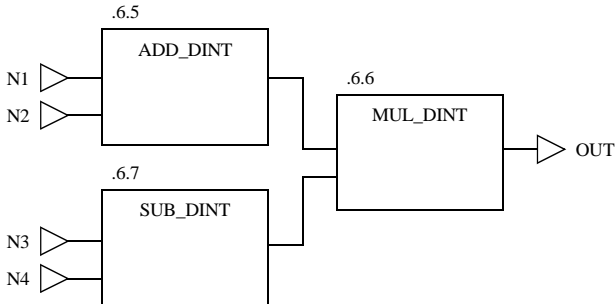
Step	Action
1	Close Concept and start Concept DFB.
2	Create a new DFB using the menu command Data file → New DFB . Reaction: The name now appears on the title bar:[untitled].
3	Using the menu command Data file → New section... , generate a new section and enter a section name. The section name (max. 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 → Preferences → IEC Extensions... → IEC Extensions → Allow leading digits in identifiers .
4	Select a programming language for the section: <ul style="list-style-type: none"> • Function Block language (FBD) (See <i>Creating the Logic in FBD Function Block Language</i>, p. 439) • Ladder diagram (LD) (See <i>Creating the Logic in LD Ladder Diagram</i>, p. 441) • Instruction list (IL) (See <i>Creating the Logic in IL Instruction List</i>, p. 445) • Structured text (ST) (See <i>Creating the Logic in ST Structured Text</i>, p. 447)
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 <i>Global / Local DFBs</i> , p. 420).
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	<p>To insert an FFB into the section, select the Objects → Select FFB... menu command.</p> <p>Result: The FFB dialog box from the library is opened.</p> <div><div>FFBs in IEC Library</div><table><tr><th>Group</th><th>EFB Type</th><th>DFB Type</th></tr><tr><td>Arithmetic</td><td>AND_BOOL</td><td>LIGHTS</td></tr><tr><td>Bistable</td><td>AND_BYTE</td><td>NEST1</td></tr><tr><td>Comparison</td><td>AND_WORD</td><td>NEST2</td></tr><tr><td>Converter</td><td>NOT_BOOL</td><td></td></tr><tr><td>Counter</td><td>NOT_BYTE</td><td></td></tr><tr><td>Edge detection</td><td>NOT_WORD</td><td></td></tr><tr><td>Logic</td><td>OR_BOOL</td><td></td></tr><tr><td>Numerical</td><td>OR_BYTE</td><td></td></tr></table><div><div>FFB sorted...</div><div>Library...</div><div>DFB</div><div>Close</div><div>Help on Type</div><div>Help</div></div></div>	Group	EFB Type	DFB Type	Arithmetic	AND_BOOL	LIGHTS	Bistable	AND_BYTE	NEST1	Comparison	AND_WORD	NEST2	Converter	NOT_BOOL		Counter	NOT_BYTE		Edge detection	NOT_WORD		Logic	OR_BOOL		Numerical	OR_BYTE	
Group	EFB Type	DFB Type																										
Arithmetic	AND_BOOL	LIGHTS																										
Bistable	AND_BYTE	NEST1																										
Comparison	AND_WORD	NEST2																										
Converter	NOT_BOOL																											
Counter	NOT_BYTE																											
Edge detection	NOT_WORD																											
Logic	OR_BOOL																											
Numerical	OR_BYTE																											
2	<p>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.</p>																											
3	<p>Place the selected FFB in the section.</p>																											
4	<p>When all FFBs have been positioned, close the dialog box with OK</p>																											
5	<p>Activate select mode with Objects → Select Mode, click on the FFB and move the FFBs to the desired position.</p>																											
6	<p>Activate the link mode with Objects → Link and connect the FFBs.</p> <p>For example:</p> <div><p>.6.5</p><div>ADD_DINT</div><p>.6.6</p><div>MUL_DINT</div><p>.6.7</p><div>SUB_DINT</div></div>																											

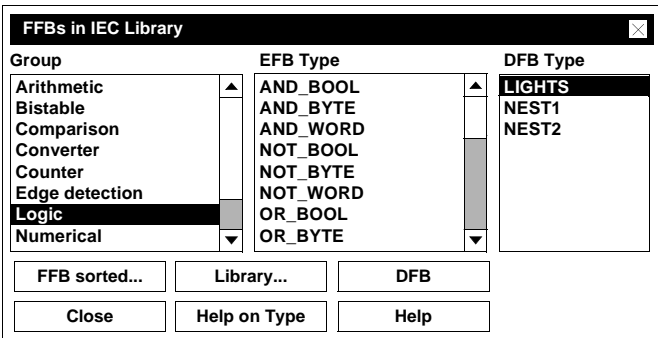
Step	Action
7	<p>Activate the Variables Editor with Project → Variable Editor to declare the DFB variables and inputs/outputs (formal parameters).</p> <p>Example (inputs):</p>  <p>Example (outputs):</p> 
8	<p>Then re-activate the select mode with Objects → Select Mode and double-click on one of the unconnected inputs/outputs.</p> <p>Result: The Connect FFB dialog box opens, in which you can allocate a current parameter to the input/output.</p>

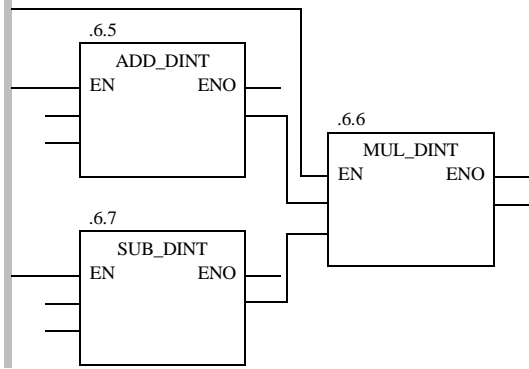
Step	Action
9	<p>Back up the DFB with the File → Save menu command.</p> <p>For example:</p>  <pre> graph LR N1 --> ADD_DINT N2 --> ADD_DINT N3 --> SUB_DINT N4 --> SUB_DINT ADD_DINT -- ".6.5" --> MUL_DINT SUB_DINT -- ".6.7" --> MUL_DINT MUL_DINT --> OUT </pre>

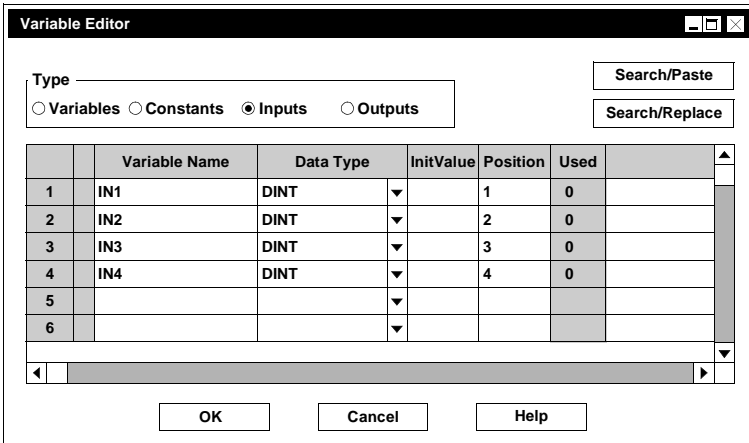
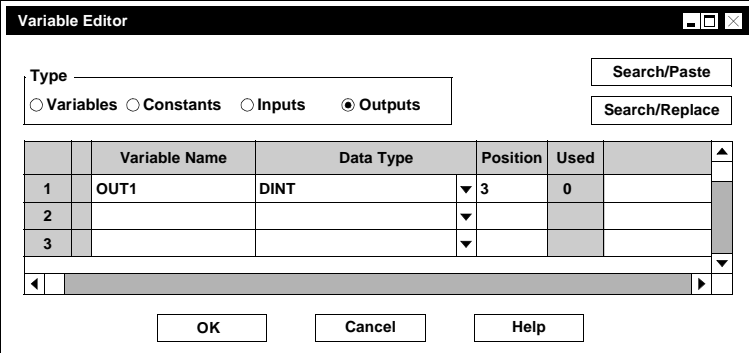
Creating the Logic in LD Ladder Diagram

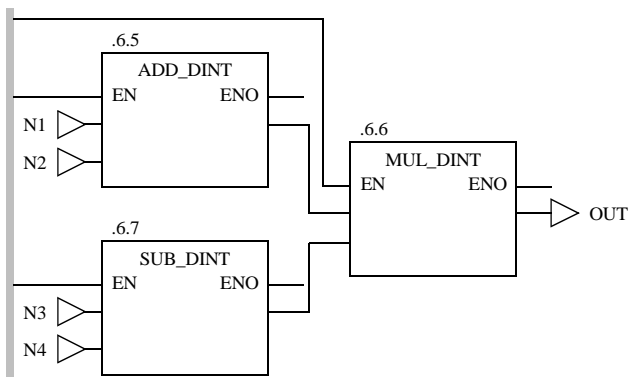
Description

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	<p>To insert an FFB into the section, select the Objects → Select FFB... menu command.</p> <p>Result: The FFBs from Library dialog box is opened.</p> 
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.

Step	Action
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.
7	<p>Activate link mode with Objects → Link, and connect the contacts, coils and FFBs. Connect the contacts, FFBs and the left power rail.</p> <p>For example:</p> 

Step	Action
8	<p>Activate the Variables Editor with Project → Variable Editor to declare the DFB variables and inputs/outputs (formal parameters).</p> <p>Example (inputs):</p>  <p>Example (outputs):</p> 
9	<p>Then re-activate select mode with Objects → Select mode, and double-click on a contact or coil.</p> <p>Result: The Properties: LD Objects dialog box is opened, in which you can allocate an actual parameter to the contact/coil.</p>
10	<p>To connect the FFB input/outputs to the current parameters, double-click on one of the unconnected input/outputs.</p> <p>Result: The Connect FFB dialog box is opened, in which you can allocate a current parameter to the input/output.</p>

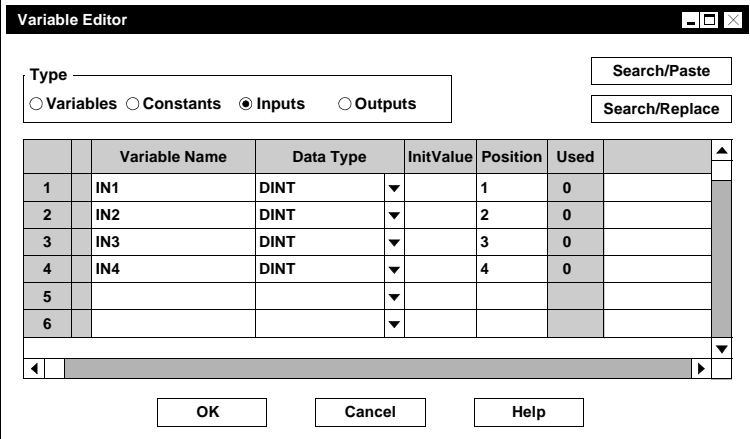
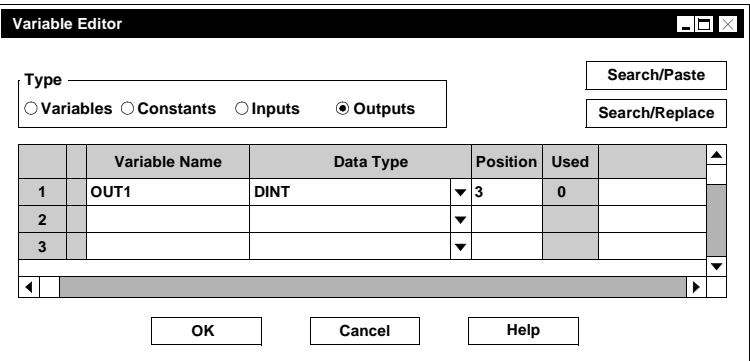
Step	Action
11	<p>Back up the DFB with the File → Save menu command.</p> <p>For example:</p>  <pre> graph LR N1((N1)) --> EN1[EN] N2((N2)) --> EN1 N3((N3)) --> EN2[EN] N4((N4)) --> EN2 subgraph DFB_65 [".6.5 ADD_DINT"] EN1 ENO1[ENO] end subgraph DFB_67 [".6.7 SUB_DINT"] EN2 ENO2[ENO] end subgraph DFB_66 [".6.6 MUL_DINT"] EN3[EN] ENO3[ENO] end ENO1 --> EN3 ENO2 --> EN3 ENO3 --> OUT((OUT)) </pre>

Creating the Logic in IL Instruction List

Description

The procedure for creating the logic in Instruction List (IL) is as follows:

Step	Action
1	<p>Declare the function block and DFBs to be used using VAREND_VAR.</p> <p>Note: Functions do not have to be declared:</p> <p>Example:</p> <pre>VAR CLOCK : CLOCK_DINT ; END_VAR</pre>

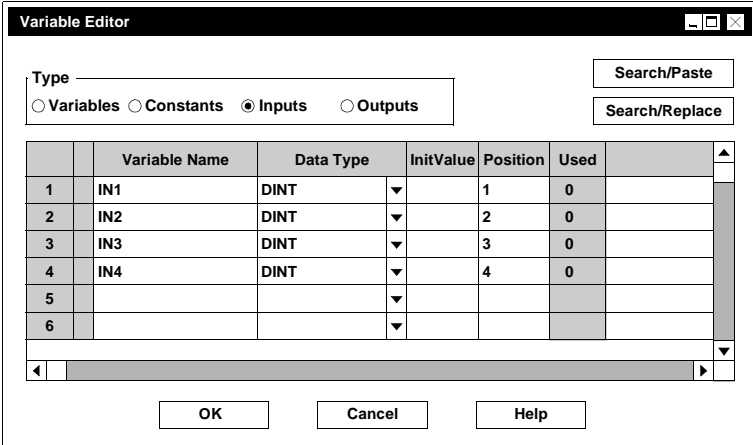
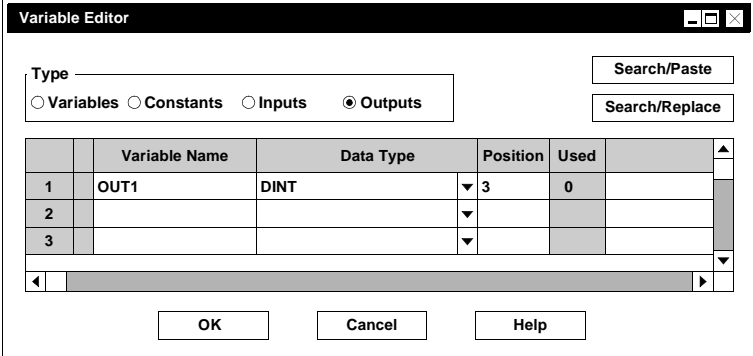
Step	Action
2	<p>Declare the variables and their initial value in the Variable Editor.</p> <p>Example (inputs):</p>  <p>Example (outputs):</p> 
3	<p>Create your program's logic.</p> <p>For example:</p> <pre>LD IN1 ADD IN2 MUL (LD IN3 SUB IN4) ST OUT</pre>
4	Back up the section with the File → Save Project menu command.

Creating the Logic in ST Structured Text

Description

The procedure for creating the logic in ST structured text is as follows:

Step	Action
1	<p>Declare the function block and DFBs to be used using VAREND_VAR.</p> <p>Note: Functions do not have to be declared:</p> <p>Example:</p> <pre>VAR CLOCK : CLOCK_DINT ; END_VAR</pre>

Step	Action
2	<p>Declare the variables and their initial value in the Variable Editor.</p> <p>Example (inputs):</p>  <p>Example (outputs):</p> 
3	<p>Create your program's logic.</p> <p>For example:</p> $OUT := (IN1 + IN2) * (IN3 - IN4)$
4	Back up the section with the File → Save Project menu command.

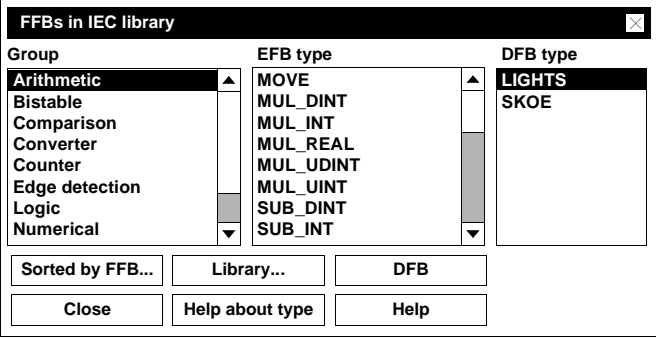
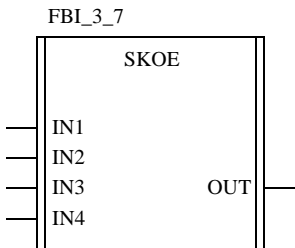
Calling up a DFB in the FBD Function Block dialog

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: 
5	Now click on the desired DFB in the list, and position it in the Editor window. For example: 
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.

Step	Action
7	<p>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).</p> <p>For example:</p> <p>The diagram illustrates two instances of the SKOE DFB. The first instance, SKOE1, has four inputs: VALUE1, VALUE2, VALUE3, and VALUE4, which are connected to the internal inputs IN1, IN2, IN3, and IN4. The output OUT of SKOE1 is connected to RESULT1. The second instance, SKOE2, has four inputs: VALUE5, VALUE6, VALUE8, and VALUE9, which are connected to the internal inputs IN1, IN2, IN3, and IN4. The output OUT of SKOE2 is connected to RESULT2.</p> <p>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.</p>

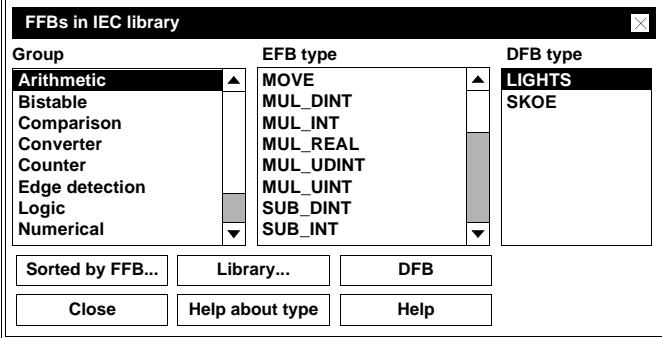
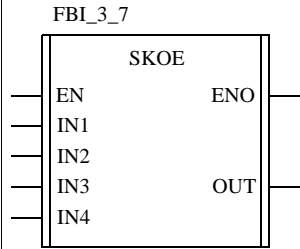
Calling up a DFB in Ladder Diagram LD

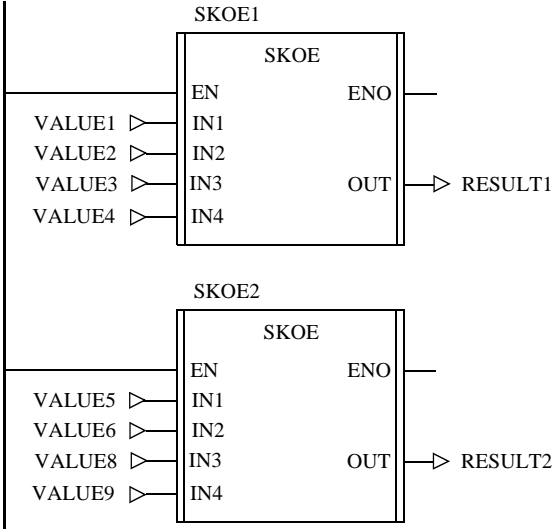
Note

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: 
5	Now click on the DFB required in the list, and position it in the Editor window. For example: 

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	<p>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).</p> <p>For example:</p>  <p>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.</p>

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	<p>Calling up a DFB in the IL is performed like Calling up a Function Block (See <i>Use of Function Blocks and DFBs</i>, p. 321).</p> <p>For example:</p> <pre> 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 </pre> <p>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.</p>

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	<p>Calling up a DFB in the ST is performed like Calling up a Function Block (See <i>Function Block/DFB Invocation</i>, p. 373).</p> <p>For example:</p> <pre>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 *)</pre> <p>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.</p>

Macros

14

At a Glance

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	457
14.2	Programming and calling up a macro	467

14.1 Macro

At a Glance

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	458
Global / Local Macros	460
Exchange marking	462
Creating Context Sensitive Help (Online Help) for Macros	465

Generals

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	<p>Macros have the following properties:</p> <ul style="list-style-type: none"> • 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 <i>Creating Context Sensitive Help (Online Help) for Macros</i> , p. 465).
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*, p. 661).

Global / Local Macros

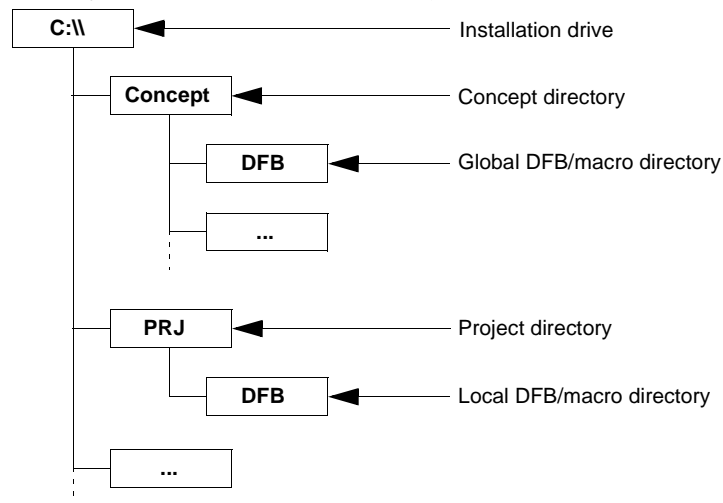
Description

Global and local macros differ in the locality of their directory hierarchy.

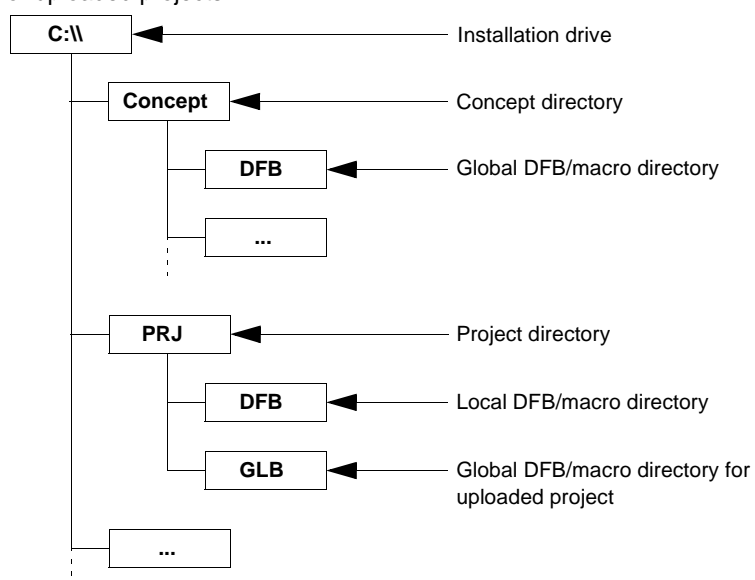
Depending on the directory or subdirectory in which the macro is stored, it can be called up globally, i.e. within all the projects created under Concept, or locally, in a specific project.

In the *Defining the Storage of Global DFBs during Upload, p. 1033* 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:



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 instantiated 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 instantancing, 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

Variable Editor

Type: ☒ Variables ☐ Constants

Find/insert

Find/replace

	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

OK Cancel Help

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	<p>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.</p>
File Format:	<p>You can create your help in the following file formats:</p> <ul style="list-style-type: none">• .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	<p>The name of the help file must be exactly the same as the name of the macro (e.g. SKOE.EXT)</p> <p>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).</p>
Directory	<p>The help file can be stored in the following directories:</p> <ul style="list-style-type: none">• 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	<p>Search for the help file MacroName.EXT in the local macro-directory. The help file is searched for in the following sequence:</p> <ul style="list-style-type: none">• .HLP• .CHM• .HTM• .RTF• .DOC• .TXT• .PDF <p>Result: If the search result is positive the help file will be displayed, otherwise it will continue with phase 2.</p>
2	<p>Search for the help file MacroName.EXT in the global macro-directory. For the order, see phase 1.</p> <p>Result: If the search result is positive the help file will be displayed, otherwise it will continue with phase 3.</p>
3	<p>Search for the help file MacroName.EXT in the Concept-directory or Concept-Help directory. For the order, see phase 1.</p> <p>Result: If the search result is positive the help file will be displayed, otherwise it will continue with phase 4.</p>
4	<p>Display of the comment created in Concept DFB with Project → Properties.</p>

14.2 Programming and calling up a macro

At a Glance

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	468
Occupying the macro	469
Creating the logic	470
Calling up a macro from an SFC section	473
Calling a macro from an FBD/LD section.	476

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 <i>Occupying the macro</i> , p. 469)
2	Creating the logic (See <i>Creating the logic</i> , p. 470)
3	Calling up the macro in: <ul style="list-style-type: none"> • Sequence language (SFC) (See <i>Calling up a macro from an SFC section</i>, p. 473) • Function Block language (FBD) (See <i>Calling a macro from an FBD/LD section.</i>, p. 476) • Ladder Diagram language (LD) (See <i>Calling a macro from an FBD/LD section.</i>, p. 476)

Occupying the macro

Description

The procedure for occupying the macro is as follows:

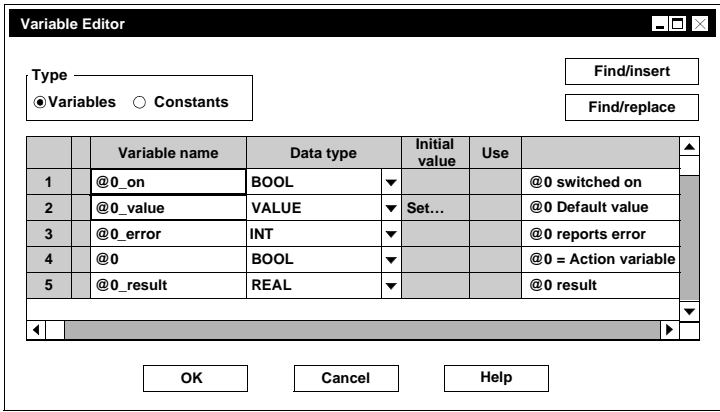
Step	Action
1	Close Concept and start Concept DFB.
2	Create a new macro using File → New macro... menu command. Reaction: The name now appears on the title bar: [untitled] .
3	Using the menu command File → New section... generate a new section and enter a section name (with an exchange marking such as @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 Options → Preferences → IEC Extensions... → IEC Extensions → Allow leading digits in identifiers .
4	Select a programming language for the section: <ul style="list-style-type: none"> • 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 <i>Global / Local Macros</i> , p. 460).
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.

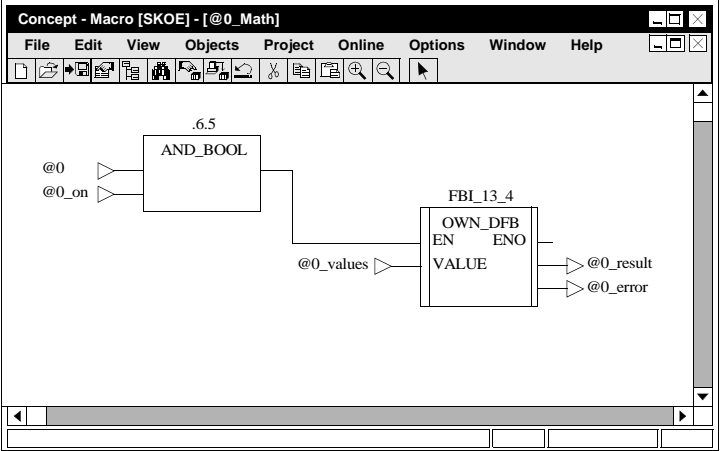
Creating the logic

Description

The procedure for creating the logic is as follows:

Step	Action																											
1	<p>To insert an FFB into the section, select the menu command Objects → Select FFB....</p> <p>Reaction: The FFBs from library dialog box opens.</p> <div><div>FFBs in IEC library</div><table><tr><th>Group</th><th>EFB type</th><th>DFB type</th></tr><tr><td>Arithmetic</td><td>MOVE</td><td>LIGHTS</td></tr><tr><td>Bistable</td><td>MUL_DINT</td><td>SKOE</td></tr><tr><td>Comparison</td><td>MUL_INT</td><td></td></tr><tr><td>Converter</td><td>MUL_REAL</td><td></td></tr><tr><td>Counter</td><td>MUL_UDINT</td><td></td></tr><tr><td>Edge detection</td><td>MUL_UINT</td><td></td></tr><tr><td>Logic</td><td>SUB_DINT</td><td></td></tr><tr><td>Numerical</td><td>SUB_INT</td><td></td></tr></table><div><div>Sorted by FFB...</div><div>Library...</div><div>DFB</div><div>Close</div><div>Help about type</div><div>Help</div></div></div>	Group	EFB type	DFB type	Arithmetic	MOVE	LIGHTS	Bistable	MUL_DINT	SKOE	Comparison	MUL_INT		Converter	MUL_REAL		Counter	MUL_UDINT		Edge detection	MUL_UINT		Logic	SUB_DINT		Numerical	SUB_INT	
Group	EFB type	DFB type																										
Arithmetic	MOVE	LIGHTS																										
Bistable	MUL_DINT	SKOE																										
Comparison	MUL_INT																											
Converter	MUL_REAL																											
Counter	MUL_UDINT																											
Edge detection	MUL_UINT																											
Logic	SUB_DINT																											
Numerical	SUB_INT																											
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 OK																											
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.																											

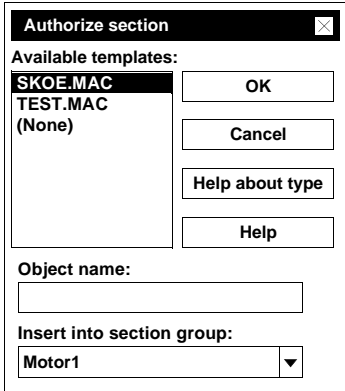
Step	Action
7	<p>Activate the Variables Editor with Project → Variables Editor to declare the variables.</p> <p>For unlocated variables, declare a name here (with exchange markings), a data type, an initial value and a comment if necessary (possibly with exchange markings).</p> <p>For constants, declare a name here (with exchange markings), a data type, a value and a comment if necessary (possibly with exchange markings).</p> <p>For example:</p>  <p>Note: 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.</p> <p>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.</p> <p>Note: 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.</p>

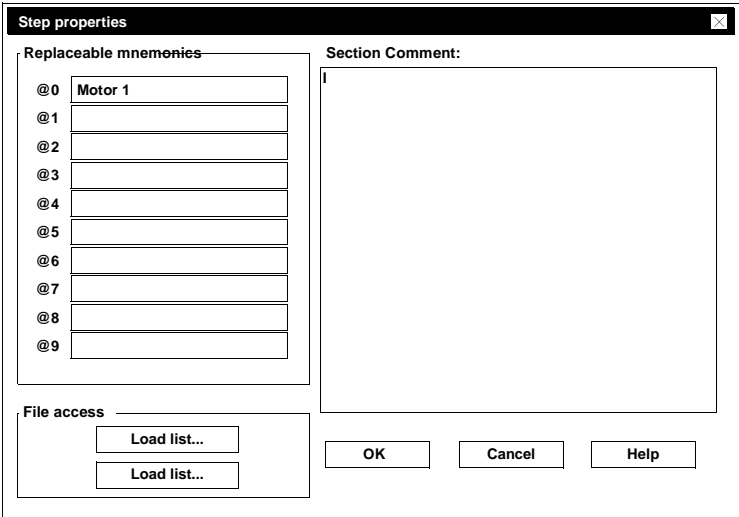
Step	Action
8	<p>Then re-activate the selection mode with Objects → Select and double-click on one of the unconnected inputs/outputs.</p> <p>Reaction: The Link FFB dialog box opens, where an actual parameter can be assigned to the input/output.</p> <div><div>Linking FFB .215 (AND_BOOL)</div><div><div>Input: IN1 (BOOL) <input type="checkbox"/> Inverted</div><div>Link with <div><input checked="" type="radio"/> Variable <input type="radio"/> Literal <input type="radio"/> Direct address</div></div><div><div>Name<div>@0_free</div><div>Look up...</div></div><div><div>Variable declaration...</div><div>OK</div><div>Cancel</div><div>Help</div></div></div></div></div>
9	<p>Save the macro with the menu command File → Save.</p> <p>For example:</p> <div><div>Concept - Macro [SKOE] - [@0_Math]</div><div><div>File Edit View Objects Project Online Options Window Help</div><div></div></div></div>

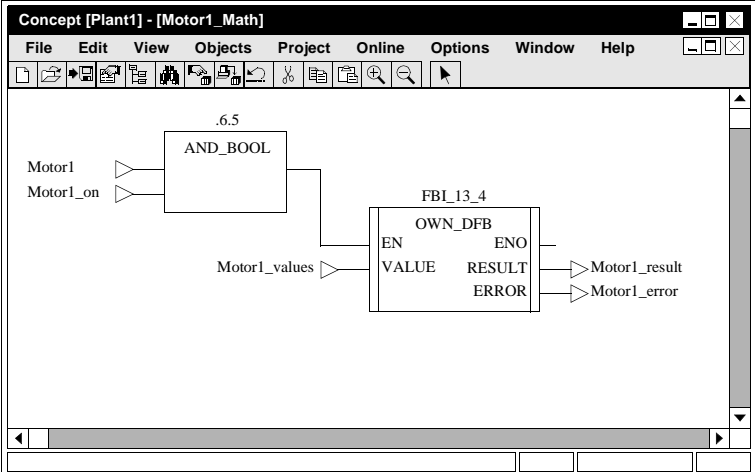
Calling up a macro from an SFC section

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	<p>Select the desired macro from the list.</p> <p>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.</p> <p>Confirm with OK.</p> <p>Example:</p>  <p>Reaction: The dialog Replace is opened to replace the exchange markings.</p>

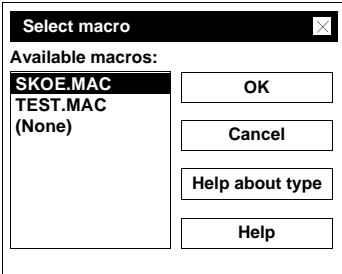
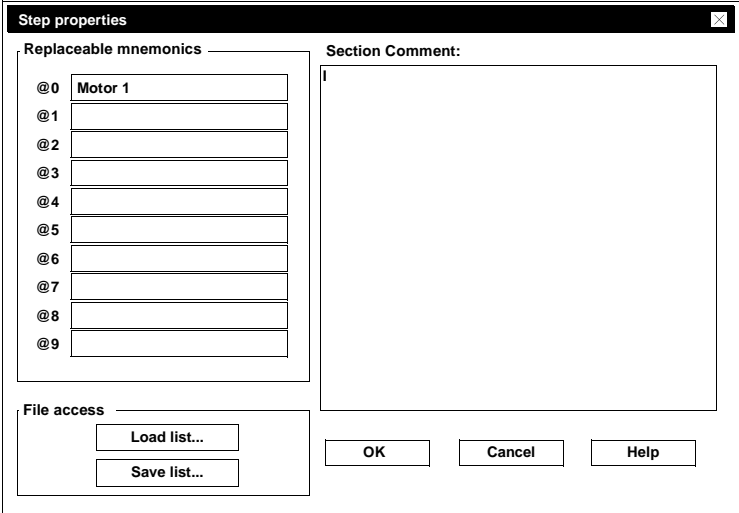
Step	Action
6	<p>Pre-set for the text fields @0 to @9 the character strings which the exchange markings are to be replaced with in the macro. Example:</p> 
7	<p>Confirm the inputs with OK.</p> <p>Reaction: The following occurs after the procedure described above has been performed:</p> <ul style="list-style-type: none"> • 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.

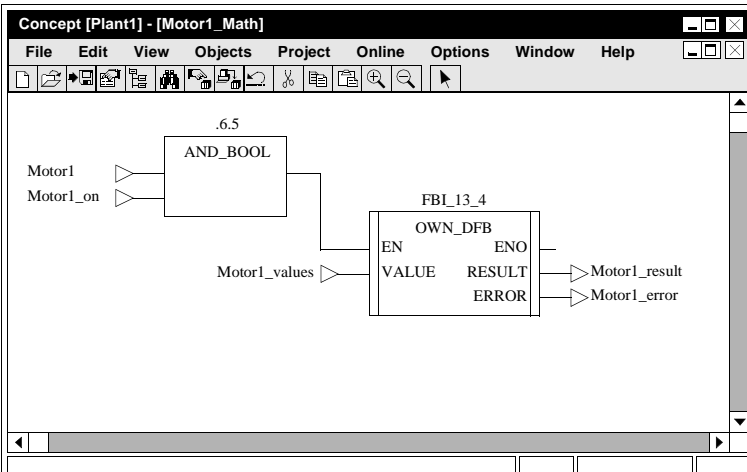
Step	Action
8	<p>This action can be used to call the macro as often as required without any name collisions occurring. The instanced macro and its variables are completely identical to the sections and variables generated beforehand.</p> <p>Example of an instanced macro:</p> 

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. 
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	Action
6	<p>Confirm the inputs with OK.</p> <p>Reaction:</p> <p>The following occurs after the procedure described above has been performed:</p> <ul style="list-style-type: none"> 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	<p>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.</p> <p>Example of an instanced macro:</p> 

Variables editor

15

At a Glance

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	480
Declare variables	480
Searching and replacing variable names and addresses	483
Searching and Pasting Variable Names and Addresses	487
Exporting located variables	490

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** → **Preferences** → **IEC Extensions...** → **IEC Extensions** → **Allow leading digits 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 → Download changes .
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 → Download changes 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	<p>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.</p> <p>Use option button Name and Address to choose whether to search for variable names or addresses.</p> <p>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.</p> <p>On activating check box Extend address the addresses specified in text box Address are automatically extended to Standard format.</p>
Use of wildcards	<p>The following wildcards can be used for searching and replacing:</p> <p>* This character is used to represent any number of characters. * can only be used at the beginning or the end of a line.</p> <p>? This character is used to represent exactly one character. If several characters are to be ignored, a certain number of ? have to be used.</p> <p>The wildcards can be combined. The combinations *? and ?* are, however, not permitted.</p>

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.

ReplaceDescription 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	<p>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.</p> <p>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.</p> <p>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.</p> <p>If you check the Extend Address check box, the addresses entered in the Address text box are automatically extended to Standard format.</p>
Using Wildcards	<p>The following wildcards can be used for searching and pasting:</p> <ul style="list-style-type: none">* 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. <p>The wild cards can be combined. The combinations *? and ?* are, however, not permitted.</p> <div><p>Note: When searching and pasting, the number of wildcards in the Search string and the Replace string has to be equal.</p></div>
Find Name	<p>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.</p>
Find Address	<p>This field is only unavailable for constants.</p> <p>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.</p>

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 NextDescription 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 PasteDescription 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

At a Glance

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	492
Detailed view in the project browser	494
Operating the Project Browser	496

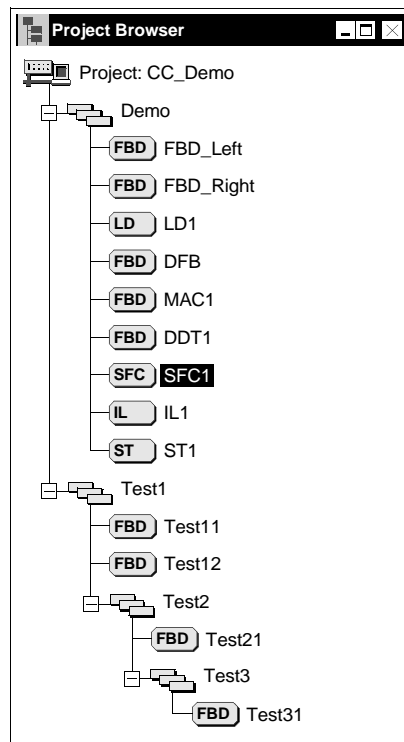
General information about the Project Browser

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

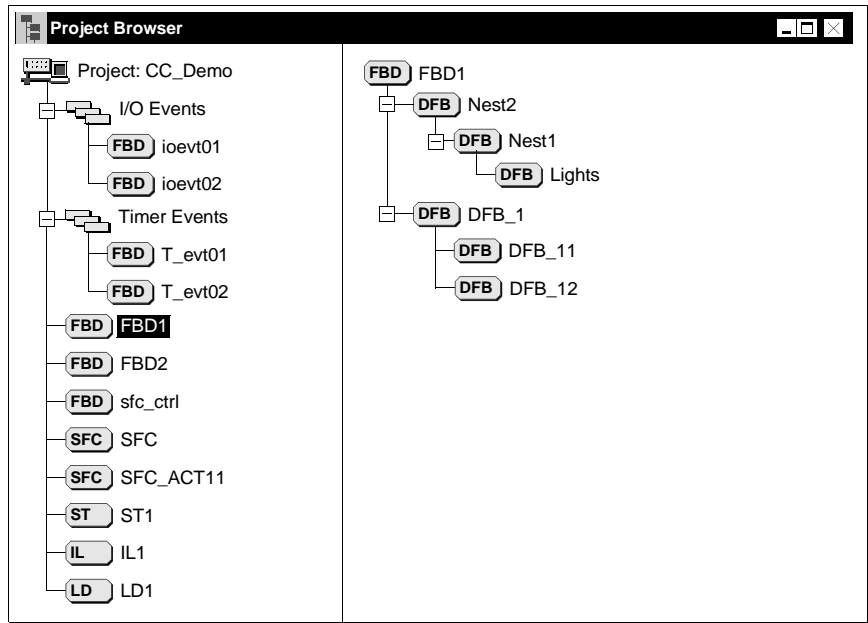
Introduction

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: <ul style="list-style-type: none">• 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

At a Glance

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	501
17.2	Syntax of the data type editor	509
17.3	Derived data types using memory	520
17.4	Calling derived data types	523

17.1 General information on Derived Data Types

At a Glance

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	502
Global / Local Derived Data Types	505
Extended Data Type Definition (larger than 64 Kbytes)	507

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.

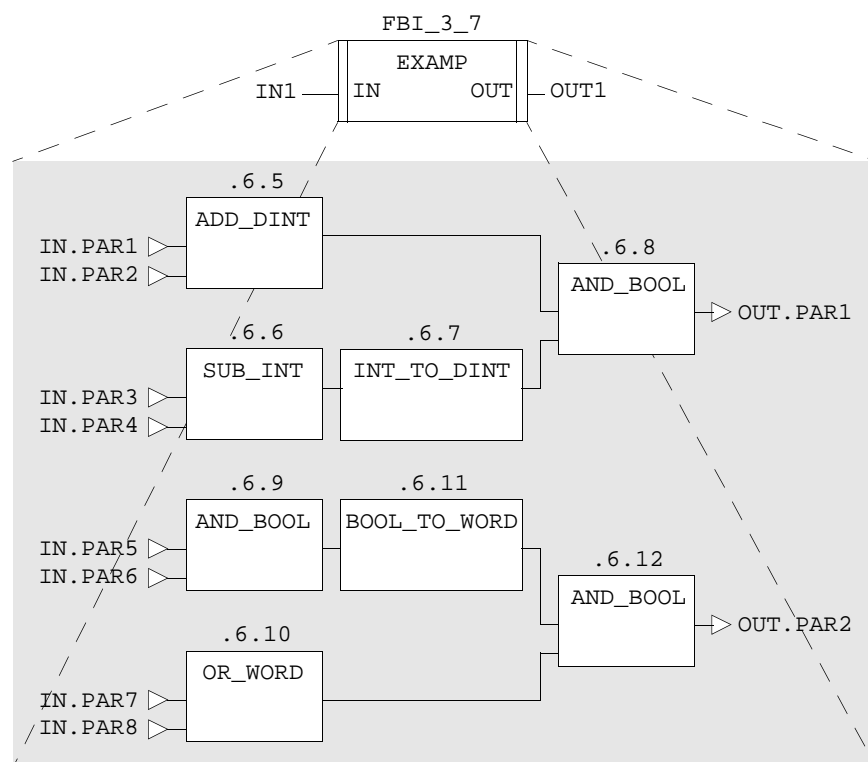
Note: Open the Data Type Editor in Concept/Concept-DFB using **File → Open → File Format Data Type Files (*.DTY)**.

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*, p. 511.

**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 *STRUCT ... END_STRUCT*, p. 512). 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 *ARRAY*, p. 513), 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

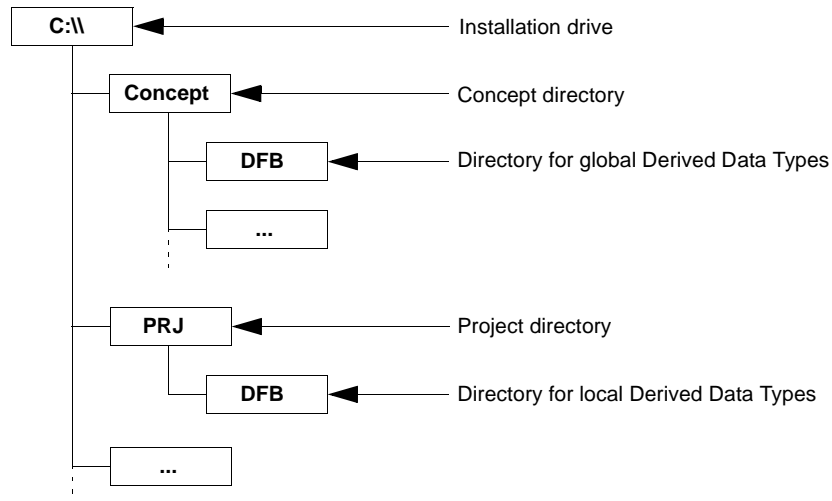
Description

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, p. 1030* 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:



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 *Elements of the Derived Data Types*, p. 510) 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

At a Glance

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	510
Key Words	512
Names of the derived datatypes	516
Separators	517
Comments	518

Elements of the Derived Data Types

At a Glance

The following elements can be used to generate the Derived Data Types:

- Key words (See *Key Words*, p. 512)
 - Names (See *Names of the derived datatypes*, p. 516)
 - Separators (See *Separators*, p. 517)
 - Comments (See *Comments*, p. 518)
-

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:

Keyword (beginning of data type definitions)

TYPE

(* Derived data type IN for EXAMP*)

Name of derived data type

IN:

Data types of structure elements

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;

Separators

Comments

Keyword (beginning of data type definitions)

(* Derived data type IN for EXAMP*)

OUT:

Name of structure elements

STRUCT

```
PAR1: DINT; (* Result of the arithmetic operations
PAR1: DINT; (* Result of the arithmetic operations
```

END_STRUCT;

Keyword (beginning of data type definitions)

Definition of array "EXP"

EXP: ARRAY [0..4] OF UINT

Keyword (end of data type definitions)

END TYPE

Key Words

Introduction

The following key words can be used to define the Derived Data Types:

- TYPE ... END_TYPE (See *TYPE ... END_TYPE*, p. 512)
- STRUCT ... END_STRUCT (See *STRUCT ... END_STRUCT*, p. 512)
- ARRAY (See *ARRAY*, p. 513)
- "Data types" (See *"Data types"*, p. 516)

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 **Options for analysis** 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 Extensions...** → **Allow leading digits 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-EPB 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 *Separator ":" (colon)*, p. 517)
 - ; (semi-colon) (See *Separator ";" (semi colon)*, p. 517)
 - [] (square brackets) (See *Separator "[]" (square brackets)*, p. 517)
 - .. (full stops) (See *Separator ".." (full stops)*, p. 518)
-

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** → **Analyze options** → **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

Storage of Boolean elements:

2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

Example of a Derived Data Type:

```

SKOE:
  STRUCT
  PAR1:  BOOL;
  PAR2:  WORD;
  PAR3:  BOOL;
  PAR4:  WORD;
  END_STRUCT;
END_TYPE

```

The diagram illustrates the bit-level structure of a 16-bit word for three different partitioning schemes. Each scheme shows a 16-bit word with bits labeled 2^{15} down to 2^0 . The bits are represented as a sequence of boxes, with the last four boxes (2^8 to 2^0) shaded gray.

- Top Scheme:** PAR2 (LSB) points to bit 2^8 , and PAR1 points to bit 2^0 .
- Middle Scheme:** PAR3 points to bit 2^8 , and PAR2 (MSB) points to bit 2^0 .
- Bottom Scheme:** PAR4 (MSB) points to bit 2^8 , and PAR4 (LSB) points to bit 2^0 .

521

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: <pre>TYPE Example1: STRUCT Par1: BOOL; Par2: INT; END_STRUCT; END_TYPE</pre>
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).
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	<p>Define two derived data types (in which the second derived data type uses the first as an element). For example:</p> <pre> 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 </pre>
2	<p>Declare a new variable in the variable editor (e.g. with the name COMPLEX_VAR).</p>
3	<p>Assign these variables the data type of the derived data type created (e.g. Example4).</p>
4	<p>Close the variable editor with OK. Reaction: A new multi-element variable called "COMPLEX_VAR" of data type "Example4" is now created.</p>
5	<p>To address this "entire" multi-element variable, simply enter the name of the variable (COMPLEX_VAR) into the program as usual.</p> <p>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]).</p>

**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 this precaution 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

At a Glance

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	532
Converting RDE templates	533
Changing signal states of a Located variable	535
Cyclical Setting of Variables	536
Unconditional locking of a section	538
Animation	539
Replacing variable names	541
Load reference data	542

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 *Converting RDE templates*, p. 533).

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
	<p>Incomplete RDE templates are created!</p> <p>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.</p> <p>Failure to follow this precaution can result in injury or equipment damage.</p>

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

At a Glance

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.

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 this precaution 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 **Download reference data**.

These settings then remain the same until the user makes a change in the RDE table, or the PLC loses the loaded data (e.g. by loading a different project).

Note: In an open RDE table, the changed date is then automatically saved using the menu command **Download reference data**. The menu command **Save template** 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.

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
	<p>Modified variable names are not recognized by replacements.</p> <p>When a variable is cyclically set, the spelling of the variable name should not be changed in the variables editor.</p> <p>Failure to follow this precaution can result in injury or equipment damage.</p>

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 .

Downloading reference data

Cyclically set values and disabled variables can be loaded onto the PLC using the menu command **Download 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 data is then automatically saved using the menu command **Download reference data**. The menu command **Save template** 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 this precaution 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 → Connect...), and load the user program onto the PLC (Online → Download...). 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 template...). 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 template...). Reaction: The RDE animation with the changed variable name is recovered.

Download 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** → **Download reference data**.

Note: To perform the loading, the animation must be disabled.
--

ASCII Message Editor

19

At a glance

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	545
19.2	User Interface of ASCII Message Editor	553
19.3	How to Continue after Getting a Warning	557
19.4	ASCII Editor in Offline/Combination/Direct Modes	559

19.1 **ASCII Editor Dialog**

At a glance

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	546
Text	547
Variables	548
Control code	549
Spaces	549
Carriage Return	550
Flush (buffer)	551
Repeat	552

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*, p. 547
- *Variables*, p. 548
- *Control code*, p. 549
- *Spaces*, p. 549
- *Carriage Return*, p. 550
- *Flush (buffer)*, p. 551
- *Repeat*, p. 552

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 $1 \leq F \leq 4$... Number of registers = N for $5 \leq F \leq 8$... Number of registers = $2 \times N$
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:

ASCII Message Editor

Message: 1

'Hello',10x,'World'

Simulation:

Hello World

Used Words: 12 Free Words: 8 Length: 12

Buttons: Delete, Delete All, View, Export, Import, OK, Cancel, Help

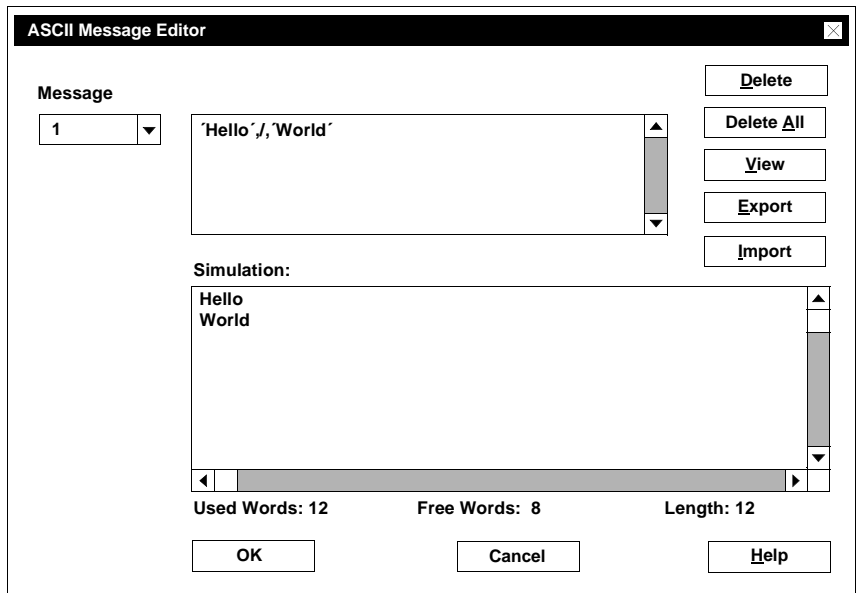
Carriage Return

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:



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: <code><0></code> clears all.
1;bbb	Removing the number of characters specified by bbb, whereby bbb is a number (1...255). For example, <code><1;100></code> 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: <code><2;5445></code> 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: <code><3;2;5445></code> 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]

Delete

Delete All

View

Export

Import

Simulation:

repeat repeat repeat

Used Words: 10

Free Words: 10

Length: 10

OK

Cancel

Help

19.2 User Interface of ASCII Message Editor

At a glance

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	554
Message Number	555
Message Text	556
Simulation Text	556

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	<p>To delete a message, select a message from the Message number list and click on Delete.</p> <p>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.</p>
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 unauthorized 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 'I'.

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

At a Glance

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	563
20.2	Connect to PLC	565
20.3	Setting up and controlling the PLC	580
20.4	Selecting Process information (status and memory)	592
20.5	Loading a project	597
20.6	Section animation	606
20.7	Online Diagnosis	610
20.8	Logging Write Access to the PLC	613

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 this precaution can result in injury or equipment damage.

20.2 Connect to PLC

At a Glance

Overview

This section contains information about connecting the PLC.

What's in this Section?

This section contains the following topics:

Topic	Page
General	566
Presettings for ONLINE operation	569
Modbus Network Link	570
Modbus Plus Network	571
Modbus Plus Bridge	576
TCP/IP-Network Link	578
Connecting IEC Simulator (32 bit)	578
State of the PLC	579

General

At a Glance

A connection can be created between a programming device and the PLC. IEC sections can be modified in Monitor mode, they cannot however, be downloaded to the PLC. When exiting Concept a warning will be displayed.

Note: Only one programming device may be connected 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:

- 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 among 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 can be found in the Concept project directory and is designated PROJECTNAME.RMK. It is for internal use only and automatically changes its content. The meaning of the individual entries can be found in the following diagram.

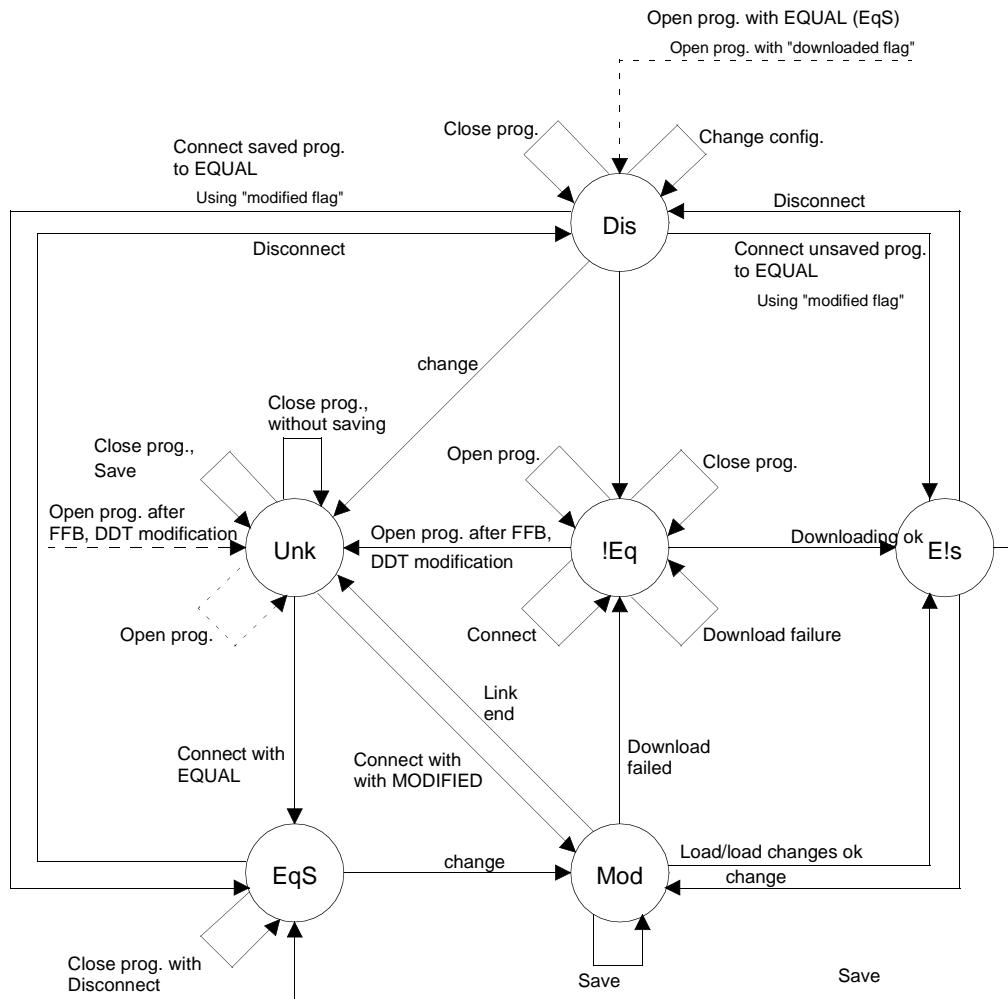
**Status
descriptions**

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...** is necessary.
 - **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 **Connect to PLC** 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 **Disconnect** 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).

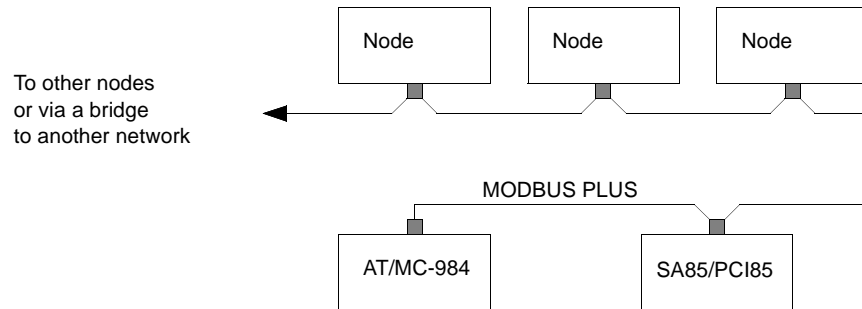
PLC as Modbus Plus Node

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.

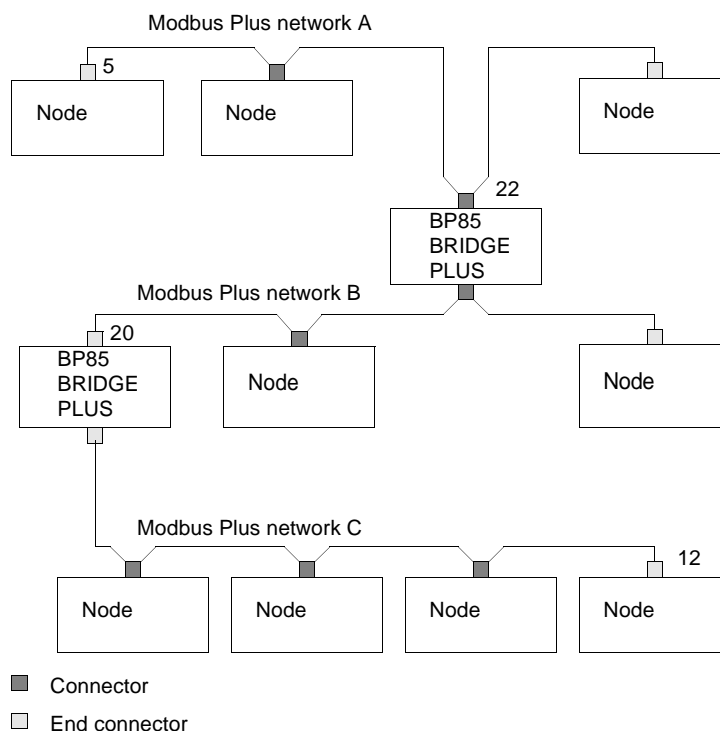
**SA85/PCI85 as
Modbus Plus
Node**

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:



Displaying a routing path with a Bridge Plus BP85:



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.

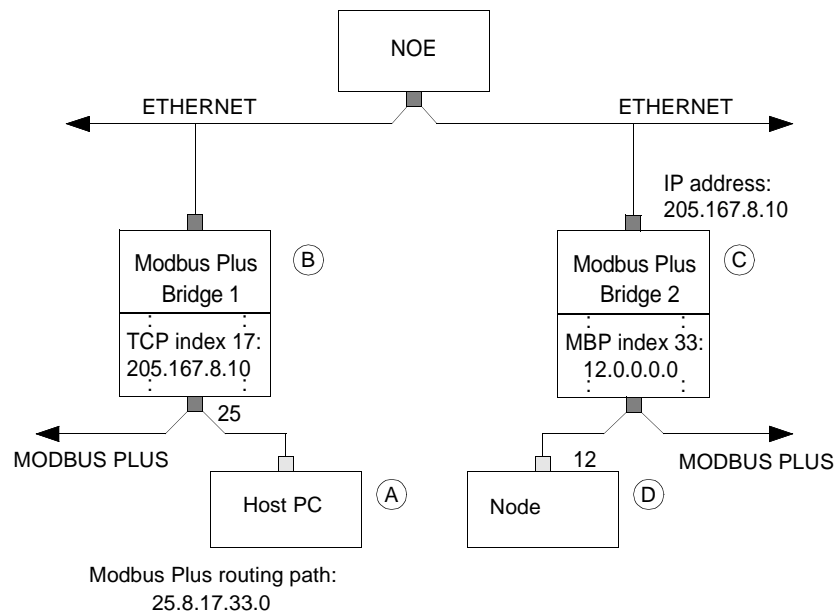
The Modbus Plus Bridge should be regarded as a host computer and must be configured in the **Protocol settings: TCP/IP** area. Enter the IP address or the host name of the Bridge, and finally in the text box **Protocol type**: change to Modbus Plus network setting.

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, p. 576".

Example:

In the dialog box **Modbus Plus Bridge** (See *Modbus Plus Bridge*, p. 576), 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.

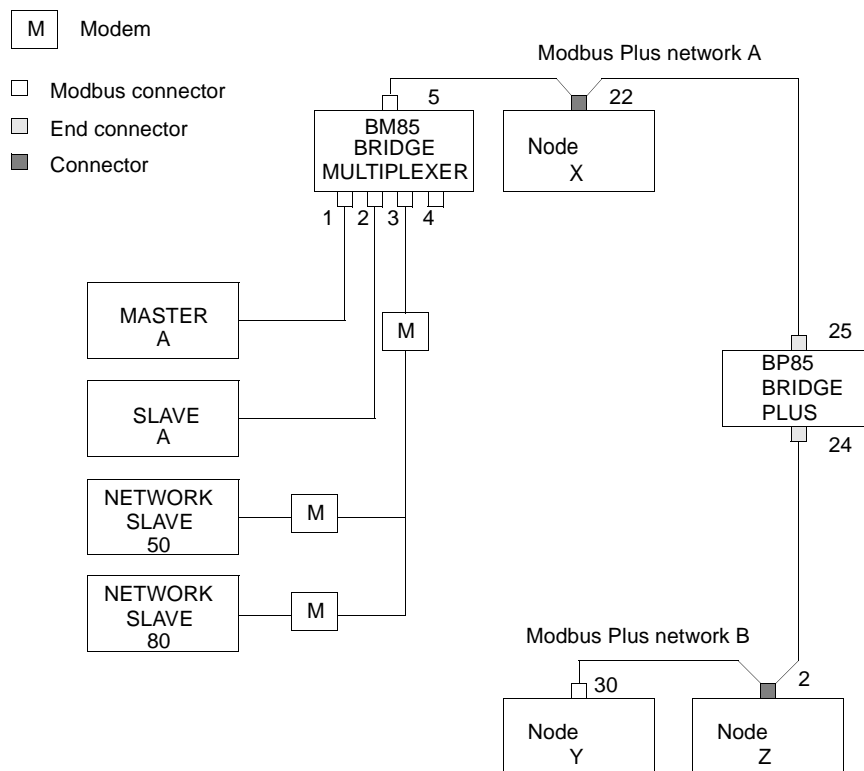
- C** Modbus Plus Bridge 2 receives the message. The MBP Index No. 3 given in 4 byte and administered by the bridge passes the message on to the configured Modbus Plus node. In this case the node 12.0.0.0.0.
- D** The message has reached its destination, Modbus Plus node 12.

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 **Connect** → **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 <i>Bridge as Modbus Plus Node</i> , p. 574)").
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 **Connect 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

At a Glance

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	581
Setting the Time for Constant Scans	582
Single Sweeps	583
Deleting memory zones from the PLC	584
Speed optimized LL984-Processing	585
Save To Flash	585
Reactivate flash save	588
Set PLC Password	589

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
	<p>It can lead to unsafe, dangerous and destructive operations of the tools or processes that are attached to the controller.</p> <p>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.</p> <p>Failure to follow this precaution can result in death, serious injury, or equipment damage.</p>

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 → Clear controller... → Clear Controller dialog box. The menu command Download... 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 panel** with the **Invoke optimized solve** command button. After the command button is activated its designation changes in **Cancel optimized solve**. 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
	<p>Modified process status after next start!</p> <p>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.</p> <p>Failure to follow this precaution can result in death, serious injury, or equipment damage.</p>

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 Start 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.
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 **Start 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 **Start 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*, p. 588 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 **Start 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 be 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 or a Momentum Ethernet CPU.

Note

The following passwords can be assigned in Concept:

- PLC password
- Concept Password (See *Changing Passwords*, p. 701) (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 <i>Set new PLC password, p. 590.</i>

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 <i>Set new PLC password, p. 590.</i>

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)

At a Glance

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	593
PLC state	594
Memory Statistics	595

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

Introduction

An overview of the memory data for the open project is given in the **Memory statistics** dialog box. The current scan time is also displayed if a real PLC is used (and not the simulator).

Total IEC memory

The memory statistics cover the following information:

Total IEC memory	Meaning
Configured	The displayed value corresponds to the value specified in the PLC Selection dialog. Note: If you use a simulator, the total memory is not given.
Used	The displayed value corresponds to that of the IEC program memory space used by the user program. Note: If you use a simulator, the used memory space is not given.

Modifying Total IEC Memory Size

The total IEC memory consists of the IEC program memory and the global data. Additional space is required in the total IEC memory for program extensions and for the administration of program modifications. The general recommendation is to set the value so that 20-30% of the value entered in the **Used** text box also remains free.

Note: Changes can only be made offline and are only accepted once the program has been downloaded to the PLC.

IEC Program Memory

The values displayed correspond to the memory space used for

- Program code
- EFB code
- Program data (section and DFB instance data)

Global Data

The memory statistics cover the following information:

Memory space	Meaning
Configured	The displayed value corresponds to the value specified in the memory space for unlocated variables in the PLC Selection dialog.
Used	The displayed value corresponds to the value from the memory space for the declared unlocated variables used by the user program.

Changing the Memory Size for Global Data

You can change the memory size of the global data. It should be noted that an increase in the global data size decreases the IEC program memory size. Each object, e.g. FFB instance, variable, step etc., takes up several bytes in the IEC program memory.
Because more memory space is not automatically gained by deleting unlocated variables, it is recommended that sufficient memory space is planned. The general recommendation is to set the value so that 20-30% of the value entered in the **Used** text box also remains free.

Note: Changes can only be made offline and are only accepted once the program has been downloaded to the PLC.
--

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 specified. After initialization, the scan time is calculated as an average value.

Note: If you are using the simulator, the scan time is not given. The display na means "not available".

20.5 Loading a project

At a Glance

Overview

This chapter contains information about loading a project.

What's in this Section?

This section contains the following topics:

Topic	Page
General information	598
Loading	599
Download Changes	600
Uploading the PLC	603
Upload Procedure	604

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 **Download 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 **Download...** 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 **All** 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.

Downloading

At a Glance

With command button **Download...** The configuration, the entire user program (IEC or LL984 section) ASCII messages (only with Concept for Quantum) and the state RAM with the initial values of a project can be sent in 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 privilege has 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	This option sends the State RAM to the PLC.
Only initial values	This option sends only initial values from the user program to the PLC. The initial values can only be loaded together with the State RAM. This means that the check box is only available when loading the State RAM.
Extended memory	This option assigns the PLC an extended memory allocation (6x-Referenzen) zu. Note: This function is only available when using Concept for Quantum.

Loading IEC selection information

To receive a complete project when selecting from the PLC **Options for code generation** the check box **IEC selection 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 problem is not possible due to insufficient program data memory, the memory size can be optimized. *Main structure of PLC Memory and optimization of memory, p. 115.*

<p>Note: If, while loading the program, a warning appears due to inconsistent DFB versions, use the menu command Project → Synchronize nested DFB versions.</p>
--

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 mode. 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.

Changes, which have no affect on the logic of the program (e.g. renaming a step name, renaming a section, renaming a variable, graphic move of a module etc.) are not loaded into the PLC with the **Download changes** function. If non-logic related changes are to be loaded into the PLC (so that, for example, these changes are available after the PLC has been uploaded to the PC), the entire project must be loaded into the PLC with **Online** → **Download**. Only then are these changes available after PLC uploading.)

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

<p>Note: If a warning appears when the program is being downloaded, due to inconsistent DFB versions, execute the menu command Project → Synchronize versions of nested DFB.</p>

ID for specific sections

The following sections contain additional ID information as they are different from cyclically set sections:


- **E** for "Event Section" (I/O Event and Timer Event Section = Interrupt-Section)
- **T** for "Transition 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
	Danger of unwanted and dangerous process conditions
	<p>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.</p> <p>Failure to follow this precaution can result in injury or equipment damage.</p>

Modified Initial values

Modified initial values are no longer loaded onto the PLC. The initial value, which was transferred to the PLC during the first load (**Download.../Download changes...**), cannot be overwritten by the **Download changes...** menu command. 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 panel → Stop controller .
2	Select the segment(s) which must be downloaded from the list.
3	Confirm using OK .
4	Call up 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 control panel → Start controller .

Loading IEC upload information

If, in the **Code generation options** dialog box, 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 *Memory Statistics*, p. 595).

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 this precaution can result in injury or equipment damage.

Uploading the PLC

Introduction

With command button **Upload...** the configuration, the entire user program (IEC or LL984 section), 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 selection is based on removing the memory, whereby this is not always compatible with the 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 way, 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.
state RAM	This option transfers the state RAM to the host computer.
Initial values only	This option only transfers initial values from the user program to the host computer. The initial values can only be uploaded together with the state RAM, i.e. the check box is only available when the State RAM is activated to upload.
Extended 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	<p>You can create an empty project using the File → New project menu command. Then execute the Online → Upload... menu command.</p> <p>Result: The Upload to project dialog is opened. Here you can determine (e.g. D:\NEWTESTPRJ.PRJ) where the project will be uploaded to.</p> <p>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.</p>
2	<p>Using the File → Open... menu command you can create a new project (e.g. D:\NEWTESTPRJ.PRJ) Then execute the Online → Upload... menu command.</p> <p>Result: The Upload Controller dialog is opened.</p>
3	<p>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.</p> <p>Result: The Upload to project dialog is opened. Here you can determine (e.g. D:\NEWTESTPRJ.PRJ) where the project will be uploaded to.</p> <p>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.</p>

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

At a Glance

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	607
LL984 Programming Modes	609

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 **Download 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 consistent. Otherwise, establish consistency using **Online** → **Download...** (if mode **UNEQUAL**) or **Online** → **Download 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** → **Connect...** .
-

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 labeled **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 original 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	<p>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.</p> <p>The diagnostics viewer only works with the SFC, FBD and LD programming languages and with the diagnostics blocks of the EXTENDED group.</p>
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	<p>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.</p> <p>A diagnostics error buffer overflow is displayed in the dialog status line.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">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.</div>
Transition Diagnosis	Information on this can be found in the <i>Transition diagnosis, p. 277</i> 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], p. 1036*)

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:

Online

- ☐ Connect to controller at startup
- ☒ Save project after download
- ☒ Online backup

OK Cancel Help

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:

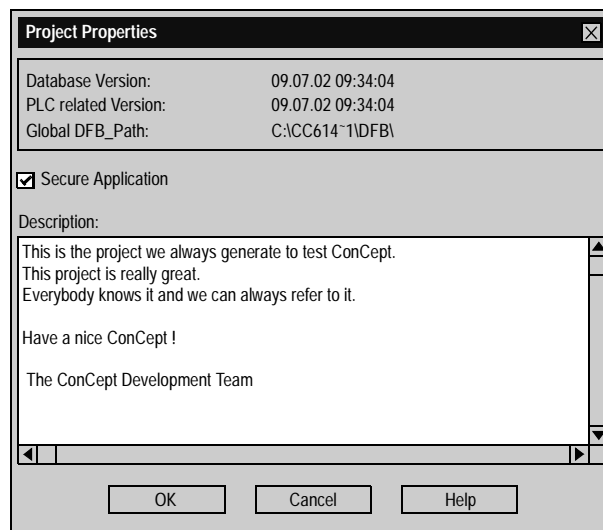
- With menu command **Options** → **Preferences** → **Common** → **Common Preferences** and the activation of the check box **Encrypt Logfile**.

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**:



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

At a Glance

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	621
21.2	Exporting sections	624
21.3	Exporting variables and derived data types	628
21.4	Section import	630
21.5	Variables import	649
21.6	Import/Export of PLC Configuration	657

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	<ul style="list-style-type: none">● 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...	<ul style="list-style-type: none">● 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	<ul style="list-style-type: none"> • 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...	<ul style="list-style-type: none"> • 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.

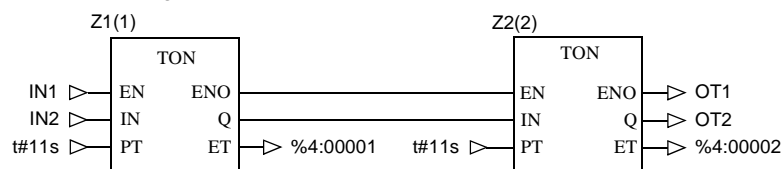
If more than one section is being exported, the code separation is expressed as an artificial PLC frame, which is not a component of the original program. It only has one INITIAL_STEP for all sections linked to it as actions (with the identifier N). These actions (sections) are executed every time the 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.

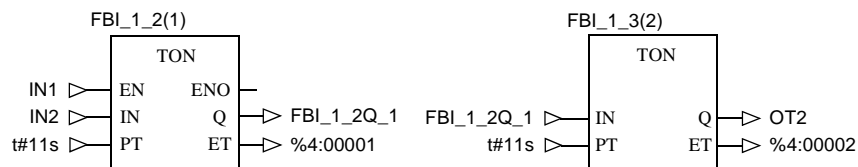
The ASCII file can be re-imported into a FBD or SFC section using the IEC text import. Using exports and imports it is possible, for example, to convert a LD section into a FBD section. Importing into a LD section is not possible.

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*, p. 645 and *Procedure for "Copying" an IL section from an existing project into a new project.*, p. 646)) 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*, p. 650 and *Importing structured variables*, p. 653).

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*, p. 650).

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 *Importing variables in Factory Link format*, p. 656).

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

At a Glance

Overview This section describes the import of sections.

What's in this Section? This section contains the following topics:

Topic	Page
Importing Sections	631
Procedure for importing sections	635
Importing IL and ST Programs to FBD, SFC, IL or ST Sections (with Conversion)	642
Importing (insert file) IL and ST programs into IL or ST sections	645
Procedure for "Copying" an IL section from an existing project into a new project.	646
Procedure for converting FBD sections from an existing project into IL sections of a new project.	647

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*, p. 658).
- 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
*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.

Step	Action
5	<p>In Save section export under , specify the name of the export file (*.SEC), e.g. C:\TARGET_DIR\TARGET.SEC</p> <p>Reaction: The sections are exported and saved in the *.SEC file, e.g. in TARGET.SEC.</p> <p>The question Import section into project now ? follows</p>
6	<p>If the question as to whether the sections should be imported is answered with OK , the import will be performed now.</p> <p>Answer Cancel, if you want to start the mport later, see also procedure Resuming following canceled import (See <i>Resuming after import cancelation</i>, p. 641).</p>
7	<p>Answer the question as to whether the project should be saved first with OK.</p> <p>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.</p>
8	<p>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 <i>Specific Changes</i>, p. 632).</p>
9	<p>Select OK to continue (the whole import process is canceled if Cancel is selected).</p> <p>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</p> <ul style="list-style-type: none"> 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. <p>Errors are displayed in the messages window and have to be acknowledged.</p>
10	<p>If the import had been canceled, eliminate the cause of the cancelation and carry out the Resuming after import cancelation (See <i>Resuming after import cancelation</i>, p. 641)procedure.</p>

**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 <i>Resuming after import cancelation</i> , p. 641)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 <i>Specific Changes</i> , p. 632).
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 <ul style="list-style-type: none"> 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 <i>Resuming after import cancelation</i> , p. 641)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.
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 <i>Specific Changes</i> , p. 632).

Step	Action
14	<p>Select OK to continue (the whole import process is canceled if Cancel is selected).</p> <p>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</p> <ul style="list-style-type: none"> 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. <p>Errors are displayed in the messages window and have to be acknowledged.</p>
15	<p>If the import had been canceled, eliminate the cause of the cancel and carry out the Resuming after import cancelation (See <i>Resuming after import cancelation</i>, p. 641) procedure.</p>

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	<p>In Save section export under specify the name of the export file (*.SEC), e.g. C:\TARGET_DIR\TARGET.SEC</p> <p>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.</p>
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	<p>Respond to the question as to whether the project should firstly be saved with OK.</p> <p>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.</p>
13	<p>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 <i>Specific Changes</i>, p. 632).</p>
14	<p>Select OK to continue (the entire import process is canceled if Cancel is selected).</p> <p>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</p> <ul style="list-style-type: none">• 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. <p>Errors are displayed in the messages window and have to be acknowledged.</p>
15	<p>If the import had been canceled, eliminate the cause of the cancel and carry out the Resuming after import cancelation (See <i>Resuming after import cancelation</i>, p. 641) procedure.</p>

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	<p>Answer the question Back up project?: with Yes.</p> <p>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.</p>
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 <i>Specific Changes</i> , p. 632).
6	<p>Select OK to continue (the whole import process is canceled if Cancel is selected).</p> <p>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</p> <ul style="list-style-type: none"> • 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. <p>Errors are displayed in the messages window and have to be acknowledged.</p>

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 POU's 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 POU's:

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. POU's 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 POU's 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	<p>The following restrictions occur when changing to FBD:</p> <ul style="list-style-type: none"> • The following restrictions occur when changing to FBD: • Block items can only be called once • only assignments and block calls but none: <ul style="list-style-type: 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	<p>The following limitations should be noted when making a SFC import from a text file:</p> <ul style="list-style-type: none"> • 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. <p>The following additional restrictions occur when changing to SFC (table = IEC 1131-3-table):</p> <ul style="list-style-type: none"> • 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 <i>Exporting FBD section.</i> , p. 647).
2	Importing FBD section into an IL section (See <i>Importing FBD section into an IL section</i> , p. 648).
3	Correcting the syntax (See <i>Correcting the syntax</i> , p. 648).

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

At a Glance

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	650
Importing structured variables	653
Importing variables in Factory Link format	656

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	<p>An ASCII file in "text delimited" format must conform to the following conditions:</p> <ul style="list-style-type: none"> • 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	<p>Order of Parameters within a Line:</p> <ul style="list-style-type: none"> • 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 *Formal parameters*, p. 422) (Concept DFB only)
- 4 or O = the symbolic name refers to an Output (See *Formal parameters*, p. 422) (Concept DFB only)
- 5 or M = the symbolic name refers to a VARINOUT variable (See *Combined Input/Output Variables (VARINOUT Variables)*, p. 423) (Concept DFB only)
- S = Structured variable, see *Importing structured variables*, p. 653.

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 *Importing Variables in "Text Delimited" Format*, p. 650) 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:
 1. Variable flag
 2. Variable name (symbolic name)
 3. Name of derived data type
 4. Hardware address
 5. Empty field
 6. 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.
 1. Character "S" (S stands for structured)
 2. Path of components (the variable name does not have to be included)
 3. Field for IEC data type (this field can remain empty)
 4. Empty field
 5. Initial value
 6. 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 plugged *)
    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

21.6 Import/Export of PLC Configuration

Introduction

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	658
Import/Export of PLC Configuration using Concept Converter	659

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 (PROJEKTNAM.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

At a Glance

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	663
22.2	Managing projects, DFBs and macros	673

22.1 Documentation of projects, DFBs and macros

At a Glance

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	664
Documentation Layout	665
Defining Page Breaks for Sections	667
Use of keywords	671

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

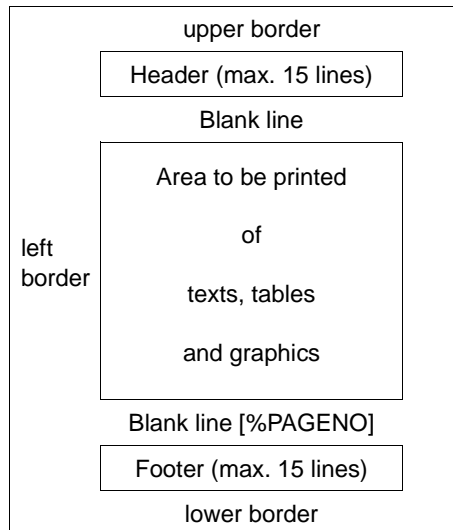
The following chapter can be printed for DFB/macro documentation using the menu command **File** → **Print** :

- 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 <i>Defining Page Breaks for Sections</i> , p. 667 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 *Use of keywords*, p. 671) 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 *Use of keywords*, p. 671) 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 *Use of keywords*, p. 671) 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

LD

70

35

From Network

Landscape

SFC

11

20

To Network

Width

Height

FBD

75

100

Load standard values

OK

Cancel

Help

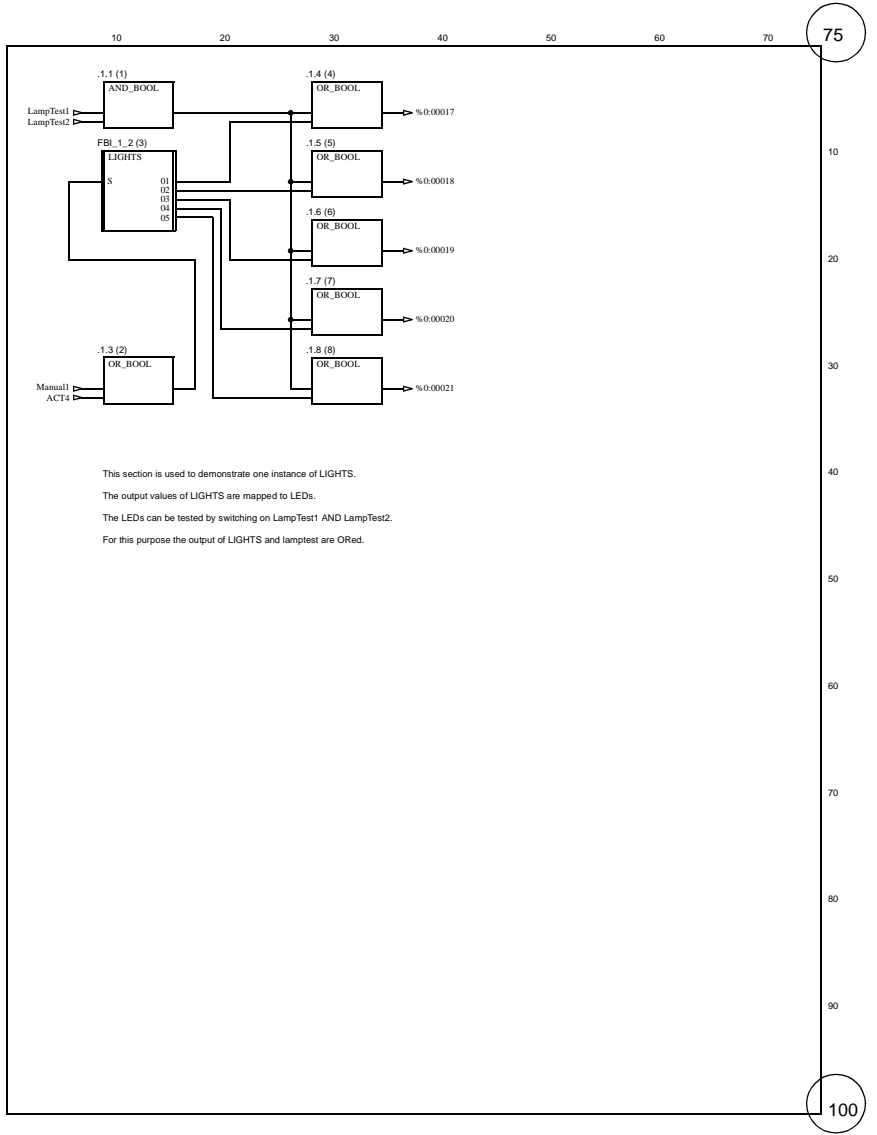
668

840 USE 503 00 October 2002

[illegible]

- 669

Print-out



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

At a Glance

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	674
Deleting projects, DFBs and macros	676

Archiving projects, used DFBs, EFBs and data type files

At a Glance

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.

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 Archiving... command cannot be selected.
2	To archive, select File → Archiving... Reaction: 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 . Reaction: A window showing all Concept projects appears.
2	Go to the list field File Type and select option Archived Projects (*.prz) . Reaction: 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.
4	Select Unpack . 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 → Connect... Reaction: The PC and PLC are found in 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.

Only the global DFBs from one directory are used or only put into 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	<p>Delete the project/DFB/macro directory (including the subdirectory "dfb").</p> <p>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 *).</p>
2	<p>Use global DFBs, and global macros in the project/DFB and if these also need to be deleted, they must be deleted separately.</p> <p>Open the subdirectory "dfb" of the Concept directory and delete all files carrying the name DFBs/macros (name *).</p>

Simulating a PLC

23

Preview

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)	679
23.2	Simulating a PLC (32-bit simulator)	682

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)

At a Glance

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	683
Simulating a PLC	685
Simulating a TCP/IP interface card in Windows 98	686
Simulating a TCP/IP interface card in Windows NT	687

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 *Simulating a TCP/IP interface card in Windows 98*, p. 686) or Simulation of a TCP/IP interface card in Windows NT (See *Simulating a TCP/IP interface card in Windows NT*, p. 687).

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 this precaution 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 this precaution 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.
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

At a Glance

Overview

This chapter describes Concept Security.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
General Description of Concept Security	692
Access Rights	694
Changing Passwords	701
Activating Access Rights	702
Protecting Projects/DFBs	702

General Description of Concept Security

At a Glance	<p>You can define access rights (See <i>Access Rights</i>, p. 694) (user definitions) using Concept Security. The access rights limit the functionality of Concept and its utilities for certain users.</p> <div>Note: The Editor LL984 cannot be protected with Concept Security.</div> <p>Projects/DFBs can be protected (See <i>Protecting Projects/DFBs</i>, p. 702) from being edited using Concept Security.</p>
Scope	<p>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.</p>
Max. number of users	<p>A maximum of 128 users can be defined.</p>
Activating Concept Security	<p>After Concept is installed, Concept Security is inactive and must be activated by the system administrator (Supervisor).</p>
The system administrator	<p>Access rights are defined and Concept Security is switched on/off by the system administrator (user name: Supervisor).</p> <p>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.</p>
Changing the access rights online	<p>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.</p>

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 can view variables online.
2	Reset SFC	The same functionality as above and also: Animation panel can be used 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

Menu commands in the main menu Edit	Minimum access rights needed
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
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

Menu commands in the main menu Online	Minimum access rights needed
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
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:

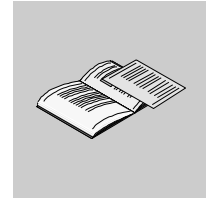
Step	Action
1	Start access management by double clicking on the Concept Security symbol.
2	Enter a user name with supervisor access rights, enter the password and confirm with OK .
3	Select the Protect register.
4	Press the command button Select and select the project/DFB to be protected. Confirm with 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



At a Glance

Overview

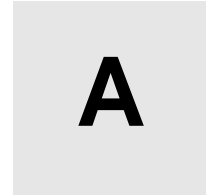
Additional information that is not necessarily required for an understanding of the documentation.

What's in this Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	Tables of PLC-dependent Performance Attributes	707
B	Windows interface	729
C	List of symbols and short cut keys	751
D	IEC conformity	779
E	Configuration examples	805
F	Convert Projects/DFBs/Macros	911
G	Concept ModConnect	915
H	Conversion of Modsoft Programs	923
I	Modsoft and 984 References	929
J	Presettings when using Modbus Plus for startup	933
K	Presettings when using Modbus for startup	947
L	Startup when using Modbus with the EXECLoader	953
M	Startup when using Modbus with DOS Loader	973
N	Startup when using Modbus Plus with the EXECLoader	987
O	Startup when using Modbus Plus with DOS Loader	1007
P	EXEC files	1023
Q	INI Files	1027
R	Interrupt Processing	1041
S	Automatic Connection to the PLC	1067

Tables of PLC-dependent Performance Attributes



Introduction

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	708
Performance Attributes of Compact	713
Performance Attributes of Momentum	717
Performance Attributes of Atrium	722

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						

Performance

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
Remote Terminal Unit (RTU) configuration extension	x	x	x	x

Performance

Performance	CPU type			
	258 (512k)	265 (512k)	275 (512k)	285 (1M)
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 Bridge Module	x	x	x
Ethernet (TCP/IP)	using Bridge Module	using Bridge Module	using Bridge Module	using Bridge Module
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
LL984 only	x	x	x	x	x
IEC only	-	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
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
Support for XMIT EFB (IEC only)	-	-	-	-	-
Support for XXMIT EFB (IEC only)	x	x	x	x	x
Upload of the user program	x	x	x	x	x
Support of Modbus function code 42 (IEC only)	-	x	x	-	x
Password protection of connection structure with PLC	-	-	-	x	x
PCMCIA support	-	-	-	-	-
Flash memory for program and configuration (LL984)	x	x	x	x	x
Flash memory for program and configuration (IEC)	-	-	x	-	x
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	-	-	-	-	-

Performance	CPU type				
	700 00	760 00	760 10	960 20	960 30
	700 10		780 10	980 20	980 30
	780 00				
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
SYMAX I/O	-	-	-	-	-
800 I/O	-	-	-	-	-
LonWorks	-	-	-	-	-
A120 I/O	-	-	-	-	-
x = available - = not available					

Buses

Availability of the buses:

Performance	CPU type				
	700 00	760 00	760 10	960 20	960 30
	700 10		780 10	980 20	980 30
	780 00				
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
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
AKFEFB (IEC only)	-	X	X	-	X
ANA_IO (IEC only)	-	X	X	-	X
COMM (IEC only)	-	-	-	-	X
CONT_CTL (IEC only)	-	X	X	-	X
DIAGNO (IEC only)	-	X	X	-	X
EXPERTS (IEC only)	-	-	-	-	X
EXTENDED (IEC only)	-	X	X	-	X
FUZZY (IEC only)	-	X	X	-	X
HANDTABLEAU (IEC only)	-	-	-	-	X
IEC (IEC only)	-	X	X	-	X
LIB984 (IEC only)	-	X	X	-	X
SYSTEM (IEC only)	-	X	X	-	X
LL984 (LL984 only)	X	X	X	X	X
x = available - = not available					

Utilities

Availability of utilities:

Performance	CPU type				
	700 00	760 00	760 10	960 20	960 30
	700 10		780 10	980 20	980 30
	780 00				
Concept DFB	-	x	x	-	x
Concept EFB	-	x	x	-	x
Concept SIM	-	x	x	-	x
Concept PLCSIM32	-	x	x	-	x
Concept security	-	x	x	-	x
Concept EXECLoader	x	x	x	x	x
Concept-Converter	x	x	x	x	x
Modsoft converter	x	x	x	x	x
Concept-ModConnect	x	x	x	x	x
x = available - = not available					

Runtime System

Runtime System

Performance	CPU type				
	700 00	760 00	760 10	960 20	960 30
	700 10		780 10	980 20	980 30
	780 00				
16 bit CPU	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	760 00	760 10	960 20	960 30
	700 10		780 10	980 20	980 30
	780 00				
Battery adapter required for backing up IEC programs	-	x	-	-	-
Floating point processor	-	-	-	-	-
Floating point emulation (IEC)	-	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	-
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

Performance

Performance	CPU type
	121 01 (2M)
	241 01 (4M)
	241 11 (4M)
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



At a Glance

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	731
B.2	Menu commands	736
B.3	Dialog boxes	739
B.4	Generating a project symbol	743
B.5	Online help	746

Windows interface

B.1 Window

At a Glance

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	732
Elements of a window	733

Window Types

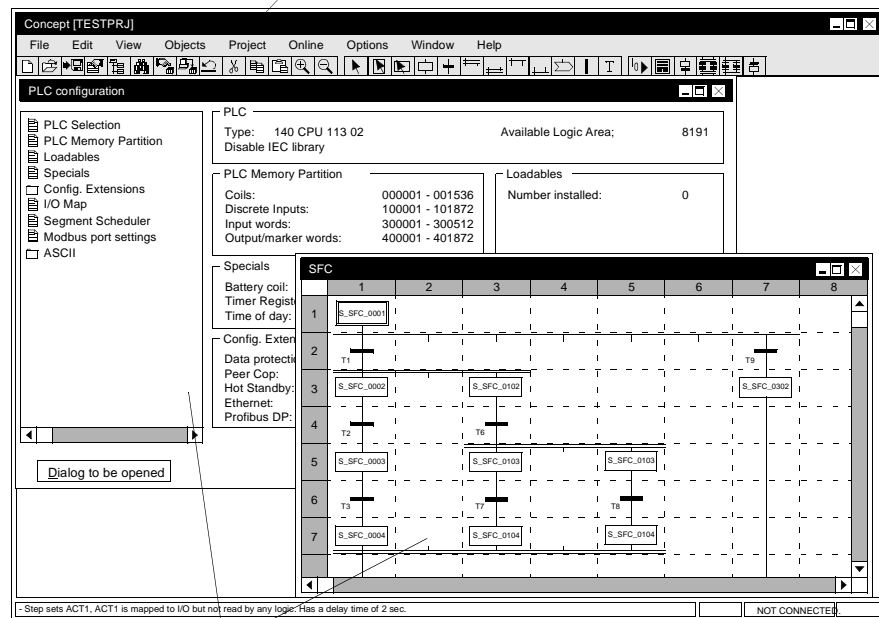
Introduction

In Windows there are two types of windows:

- Application Window
- Document Window

Types of window:

Application window (project)



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.

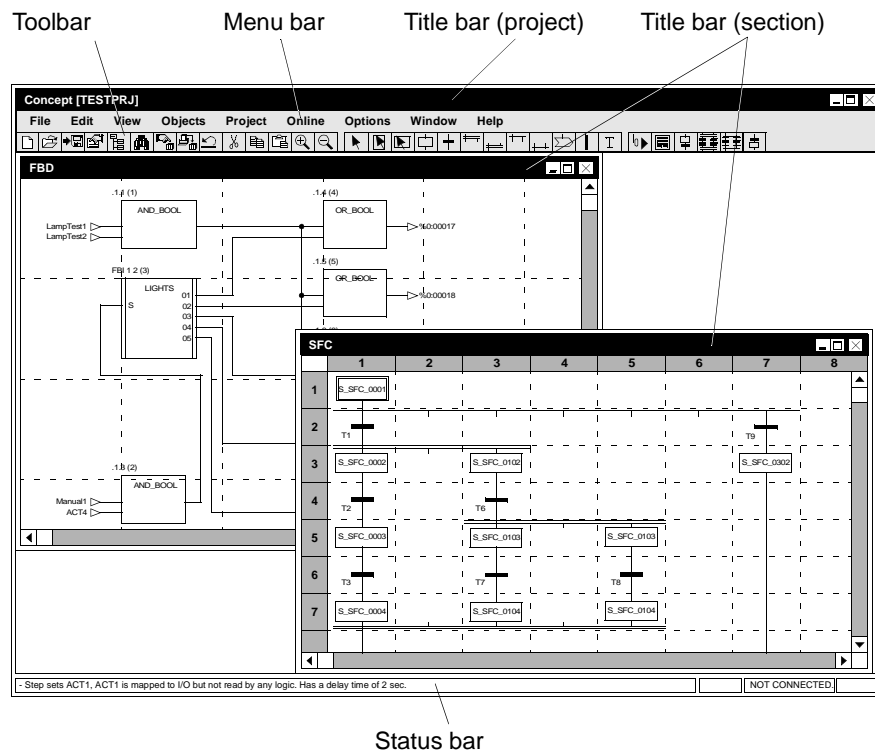
Several document windows can be open simultaneously, but only one of these can be active. An active document window can be recognized by the color of the title bar.

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:



Title bar	<p>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.</p> <p>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.</p> <p>Document windows which are not enlarged to full screen have their own title bar in which the name of the document window is indicated.</p>
Menu Bar	<p>The menu bar of the application window contains various main menus. The contents of the menu bar depend on the active document window.</p>
Toolbar	<p>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.</p> <p>There are three different ways a button can be represented:</p> <ul style="list-style-type: none">• 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	<p>The appearance of the status bar depends on whether the project is open and the programming language used in the section.</p> <p>In the first part of the status bar various information is displayed depending on the selected object.</p> <ul style="list-style-type: none">• 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** → **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**.

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:

Project	Online	Options	Window	Help
Properties...				
PLC Configuration				
Project Browser				
Execution sequence...				
Variable declaration...				
ASCII reports...				
Search...				
Trance				
Find Next				
Search results...				
References used...				
Analyze section				
Analyze program				
Options for code generation...				

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

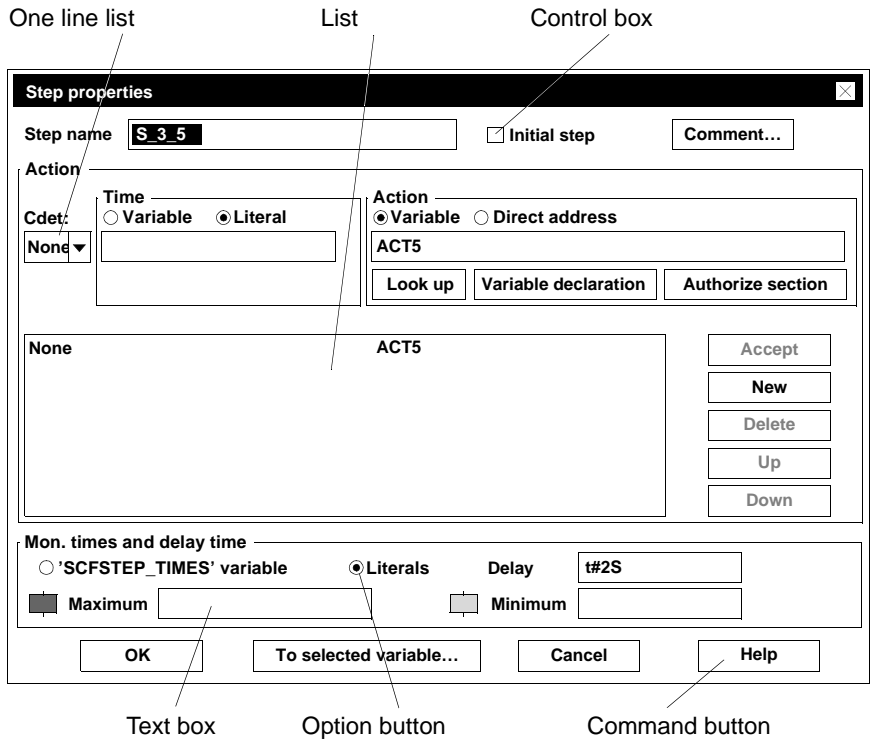
At a Glance

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:



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 from 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 <i>Automatic Connection with Command Line Parameters (Modbus, Modbus +, TCP/IP)</i> , p. 1068) 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

At a Glance

Overview This section describes use of online help.

What's in this Section? This section contains the following topics:

Topic	Page
At a Glance	747
How the Online Help is set out	748

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

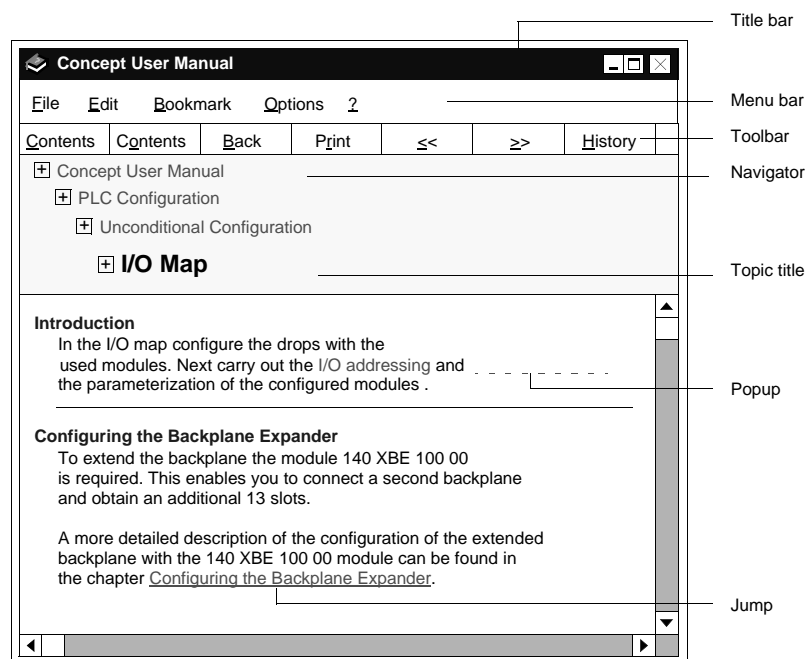
Introduction

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:



Title Bar

The title bar contains the active help file names, or in other words the help project.

Menu Bar

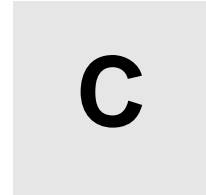
A description of the standard menu bar can be found in the respective Microsoft Windows manual.

Toolbar	<p>The following buttons are available in Concept:</p> <ul style="list-style-type: none">● 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 <i>Jump</i>, p. 749) 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 <i>Navigator</i>, p. 749).● 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	<p>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.</p>
Navigator	<p>The Navigator is in the topic title. It serves as a navigator inside the help projects.</p>
Jump	<p>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.</p>

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



At a Glance

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	753
C.2	Short cut keys	763

C.1 Icon bar

At a Glance

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	754
Icon bar in the FBD editor	755
Icon bar in the SFC-Editor	756
Icon bar in the LD editor	758
List of Symbols in the IL and ST Editor	759
List of Symbols in the LL984-Editor	760
Icons in PLC Configuration	761
Toolbar in the RDE Editor	762
Toolbar in the Project Browser	762

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 → Section open...
	File → Save
	Project → Variable declaration...
	Project → Search...
	Online → Online control panel...
	Online → Download changes...
	Edit → Reverse: 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*, p. 754):





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*, p. 754):

















Symbol	Menu entry command executed
	View → Zoom in
	View → Zoom out
	Objects → Selection
	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*, p. 754):

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
	Online → Animate selected
	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*, p. 754):

Symbol	Menu Entry Command Executed
	Objects → Insert FFB
	Online → Watch Selected
	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*, p. 754):

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 By Name
	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*, p. 754):

Icon	Executed menu command
	PLC configuration → PLC Type...
	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 Access Protection...
	PLC configuration → Peer Cop...
	PLC configuration → Ethernet / I/O Scanner...
	PLC configuration → Hot Standby...
	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*, p. 754):

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*, p. 754):

Icon	Menu command executed
	Project Shortcut Menu → Animate Enable States
	Project Shortcut Menu → Show Detailed View

C.2 Short cut keys

At a Glance

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	764
Short Cut Keys in the IL, ST and Data Type Editor	765
Short Cut Keys in the FBD and SFC Editor	768
Shortcut keys in the LD-Editor	771
Short Cut Keys in the LL984-Editor	777

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	Data file → Save project/save DFB
Alt+F4	Data 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*, p. 764):

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 → Go to line...
Ctrl+J	Edit → Go to 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*, p. 764):

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 link 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 link 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** affects the left side of the current cell

Calling up menu command

The table shows the additional shortcut keys and their corresponding menu commands available in LD Editor (see also *General Short Cut Keys*, p. 764):

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
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
N	Objects → Contact - Falling Edge
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
Shift+N	Objects → Coil - Falling Edge
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 <i>Creating objects</i> , p. 776). 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 <i>Moving the mouse pointer</i> , p. 775).
4	The mouse pointer must also be used to invert in/outputs (see <i>Moving the mouse pointer</i> , p. 775).

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	Moves the 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
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, p. 764*):

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 → Expand
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 → Go to...
Ctrl+I	Networks → Insert
Ctrl+Q	Networks → Insert Equation
Ctrl+A	Networks → Append
Ctrl+U	Networks → Append Equation
Ctrl+K	Networks → Delete
Picture down	Networks → Next
Picture up	Networks → Previous
Ctrl+M	Networks → Comment

List of symbols and short cut keys

Short Cut Keys	Menu Entry Command Executed
Ctrl+T	Online → Trace
Ctrl+B	Online → Retrace

Index



A

- Access Rights, 701, 694, 702
- Action variable, 241
- Actions, 240
 - Process, 259
- Activate dialogs, 92
- Actual parameters
 - FBD, 182
 - LD, 215
- Alias designations
 - Step, 266
 - Transition, 266
- Alternative branch, 248
- Alternative connection, 250
- Animation, 539, 679, 682
 - FBD, 193
 - General information, 606
 - IEC section, 607
 - IL, 337
 - IL/ST, 334
 - LD, 226
 - LL984 section, 609
 - Section, 606
 - SFC, 270, 272
- ANY Outputs, 373
- Archiving
 - DFB, 674
 - EFB, 674
 - Project, 674
- ARRAY
 - Range Monitoring, 527

- ASCII message editor, 543, 545, 550
 - Combination mode, 559, 560
 - Control code, 549
 - Direct mode, 559, 560
 - Flush (buffer), 551
 - Generals, 546
 - How to continue after getting a warning, 557, 558
 - How to Use, 554
 - Message Number, 555
 - Message text, 556
 - Offline mode, 559, 560
 - Repeat, 552
 - Simulation text, 556
 - Spaces, 549
 - Text, 547
 - User interface, 553, 554
 - Variables, 548
- ASCII messages, 56, 91
- Assign instructions
 - ST, 357
- Assignment
 - =>, 376
- Atrium
 - Memory optimization, 163
- Atrium example
 - INTERBUS controller, 877
- Atrium first startup
 - DOS Loader, 1014
 - EXECLoader, 996
 - Modbus Plus, 996, 1014
- Auto-Log-Out, 112

Automatic Connection, 1068, 1071
Available functions in OFFLINE and ONLINE modes, 76

B

Backplane Expander
 Configure, 98
 Edit I/O Map, 99
 Error handling, 100
 Generals, 99

Block call up
 IL, 320
 ST, 372

C

Call
 DFB, 319
 FFB, 319
 FFB, 327
 Project, 744
Chain jump, 246
Chain loop, 247
Change
 Coil, LD, 220
 Contact, LD, 220
 FFB, FBD, 187
 FFB, LD, 220
Changing signal states of a Located variable
 Reference data editor, 535
Close Column
 LL984, 398
Closer
 LD, 205
Code generation
 FBD, 191
 IL, 332
 ST, 381
 LD, 224
Coil
 Change, LD, 220
 LD, 206
 Replace, LD, 220
Coil - negated
 LD, 207

Coil – negative edge
 LD, 208
Coil – positive edge
 LD, 207
Coil - reset
 LD, 208
Coil - set
 LD, 208
Cold restart, 37
Comments
 Data type editor, 518
 Derived data type, 518
Communication, 14
Compact
 Memory optimization, 147
Compact configuration
 RTU extension, 105
Compact example, 871
Compact first startup
 DOS Loader, 977, 1011
 EXECLoader, 958, 992
 Modbus, 958, 977
 Modbus Plus, 992, 1011
Concept DFB, 415, 455
Concept ModConnect, 915
 Integrating new Modules, 919
 Removing modules, 920
 Use of Third Party Modules in Concept, 922
Concept PLCSIM32, 682
Concept Security, 691, 692, 694, 701, 702
Concept SIM, 679
CONCEPT.INI, 1027, 1029
 General, 1030
 LD section settings, 1035
 Path for Global DFBs, 1033
 Path for Help Files, 1033
 Path for MBPPATH.INI, 1033
 print settings, 1031
 Project Name Definition, 1032
 Reading Global DFBs, 1033
 Register Address Format Settings, 1032
 Representation of internal data, 1035
 Security Settings, 1037
 Setting for Online Processing, 1036
 Setting for the Address Format, 1036

- Settings for Warning Messages, 1036
- Storage of Global DFBs during Upload, 1033
- Variable Storage Settings, 1032
- Configuration, 51, 69
 - Backplane Expander Config, 98
 - Ethernet, 104
 - Ethernet I/O Scanner, 106
 - General information, 71
 - INTERBUS, 102
 - Network systems, 92
 - Optional, 90
 - OFFLINE and ONLINE mode, 74
 - Profibus DP, 103
 - RTU extension, 105
 - Unconditional, 78
 - Various network systems, 101
- Configuration example
 - Atrium-INTERBUS controller, 877
 - Compact controller, 871
 - Momentum-Ethernet bus system, 895
 - Momentum-Remote I/O bus, 887
 - Quantum-INTERBUS control, 835
 - Quantum-Peer Cop, 863
 - Quantum-Profibus DP controller, 849
 - Quantum-Remote control with DIO, 826
 - Quantum-Remote control with RIO, 807
 - Quantum-Remote control with RIO (series 800), 815
 - Quantum-SY/MAX controller, 841
- Configuration extensions, 92
- Connect
 - PLC, 565
 - Automatically with command line parameters, 1068
 - Automatically with the CCLaunch Tool, 1071
- Connect to IEC Simulator (32-bit), 578
- Constant Scan, 582
- Constants, 35
- Contact
 - Change, LD, 220
 - LD, 205, 206
 - Replace, LD, 220
- Context help, 746

- Convert
 - DFBs, 911
 - Macros, 911
 - Modsoft programs, 923
 - Projects, 911
 - RDE templates, 533
- CPU selection for the PLC type, 80
- Create
 - DFB, 436
 - FFB, FBD, 186
 - FFB, LD, 219
 - Macro, 467
 - Program, 47
 - Project, 47
 - Project Symbol, 744
- Creating a program
 - IL, 339
- Cyclical Setting of Variables
 - Reference Data Editor, 536

D

- Data exchange between nodes on the Modbus Plus network, 93
- Data flow, 221
 - FBD, 189
- Data protection, 55
- Data protection in the state RAM, 94
- Data Type Definition
 - Extended (larger than 64 Kbytes), 507
- Data type editor, 499, 501
 - Comments, 518
 - Elements, 510
 - Key words, 512
 - Names, 516
 - Separators, 517
 - Short Cut Keys, 765
 - Syntax, 509
 - Use of memory, 521
- DDT, 507
- Declaration of variables, 480
- Declare
 - Actions, 259
 - Step properties, 257
 - Transition, 264

- Defining Colors
 - INI File, 1036
- Defining the LD contact connection
 - Settings in the INI file, 1035
- Defining the number of LD columns/fields
 - Settings in the INI file, 1035
- Delete
 - DFB, 676
 - Macro, 676
 - Memory zones from the PLC, 584
 - PLC contents, 584
 - Project, 676
- Derived Data Type, 499, 501
 - Comments, 518
 - Elements, 510
 - Export, 629
 - Global, 505
 - Key words, 512
 - Local, 505
 - Names, 516
 - Separators, 517
 - Syntax, 509
 - Use of memory, 521
- Derived Data Types
 - Use, 524
- Derived Function Block, 418
 - FBD, 180
 - LD, 211
- DFB, 415, 418
 - Archiving, 674
 - Call, 319
 - Convert, 911
 - Context sensitive help, 434
 - Create, 436
 - Creating Global Variables, 430
 - Delete, 676
 - Documentation, 663
 - FBD, 180
 - Global, 420
 - Invocation, 321, 373
 - LD, 211
 - Local, 420
 - Protect, 702
- Diagnosis
 - Transition diagnosis, 277
- Diagnosis viewer, 610
- Dialog boxes, 740
- Dialog interaction
 - LL984, 394
- Direct Addresses, 35
- Disable
 - Interrupt Sections, 42
 - Section, 42
- Document section options, 667
- Documentation
 - Contents, 664
 - DFB, 663
 - Keywords, 671
 - Layout, 665
 - Macro, 663
 - Project, 663
- DOS Loader
 - Atrium first startup, 1014
 - Compact first startup, 977, 1011
 - Momentum first startup, 980, 983, 1017, 1020
 - Quantum first startup, 974, 1008
 - Startup when using Modbus, 973
 - Startup when using Modbus Plus, 1007
- Download Changes, 600
- Driver for 16 bit application capability with Windows 98/2000/NT
 - Virtual MBX Driver, 940
- Driver for connection between ModConnect Host interface adapters and 32 bit applications with Windows 98/2000/NT
 - MBX-Treiber, 941
- Driver for Modbus Plus Function via TCP/IP
 - Ethernet MBX Driver, 943
- Driver for Remote Operation
 - Remote MBX Driver, 942
- DTY, 499, 501, 502
- DX Zoom
 - LL984, 400

E

- Edit
 - Actions, 259
 - LL984, 393, 397
 - SFC, 253
 - Step properties, 257
 - Transition, 264
- Edit I/O Map
 - Backplane Expander, 99
- Editing local Drop, 808
- Editing Networks
 - LL984, 398
- Editors, 9
- EFB
 - Archiving, 674
 - FBD, 178
 - LD, 209
- EFBs for Interrupt Sections, 1065
- Elementary Function
 - FBD, 178
 - LD, 209
- Elements
 - Data type editor, 510
 - Derived Data Type, 510
- EN
 - FBD, 181
 - LD, 213
- ENC File, 15, 613, 614
- Encrypt Logfile, 15, 693
 - ENC File, 614
- ENO
 - FBD, 181
 - LD, 213
- EQUAL, 566
- Equation network
 - LL984, 404, 405
- Equation network, Syntax and Semantics
 - LL984, 409
- Error handling
 - Backplane Expander, 100
- Establishing the hardware connection
 - Modbus Plus presettings, 945
 - Modbus presettings, 950
- Ethernet, 578
- Ethernet / I/O Scanner
 - Configurator, 106
 - How to use the Ethernet / I/O Scanner, 109
- Ethernet Bus System
 - Create online connection, 910
 - Momentum, 896
- Ethernet MBX Driver
 - Driver for Modbus Plus Function via TCP/IP, 943
- Ethernet with Momentum, 105
- Ethernet with Quantum, 104
- Event Viewer
 - INI Settings, 1039
- Example of hardware configuration
 - Atrium-INTERBUS controller, 877
 - Compact controller, 871
 - Momentum-Ethernet bus system, 895
 - Momentum-Remote I/O bus, 887
 - Quantum-INTERBUS control, 835
 - Quantum-Peer Cop, 863
 - Quantum-Profibus DP controller, 849
 - Quantum-Remote control with DIO, 826
 - Quantum-Remote control with RIO, 807
 - Quantum-Remote control with RIO (Series 800), 815
 - Quantum-SY/MAX controller, 841
- Exchange Marking
 - Macro, 462
- EXEC file, 1023
 - CPU 424 02, 125
 - CPU X13 0X, 125
 - Momentum, 160
- EXECLoader
 - Atrium first startup, 996
 - Compact first startup, 958, 992
 - Momentum first startup, 962, 967, 999, 1003
 - Quantum first startup, 954, 988
 - Startup when using Modbus, 953
 - Startup when using Modbus Plus, 987
- Execution Order
 - FBD, 187
 - Section, 41
 - Timer Event Sections, 1051

- Execution sequence
 - LD, 221
- Export, 619
 - Derived Data Type, 629
 - General Information, 622
 - PLC Configuration, 658
 - Section, 625
 - Variable, 629
- Exporting located variables, 490
- Expressions
 - ST, 345
- Extended memory, 129

F

- Factory Link, 656
- FBD, 173
 - Actual parameters, 182
 - Animation, 193
 - Calling a macro, 476
 - Code generation, 191
 - Data flow, 187, 189
 - Derived Function Blocks, 180
 - DFB, 180
 - EFB, 178
 - Elementary Function, 178
 - Elementary Function Block, 179
 - EN, 181
 - ENO, 181
 - Execution order, 187
 - FFB, 178
 - Function, 178
 - Function Block, 179
 - Icon bar, 755
 - Link, 182
 - Loop, 189
 - Online Functions, 193
 - Program creation, 196
 - Short Cut Keys, 768
 - Text Object, 184
 - UDEFB, 181
 - User-defined Elementary Function, 181

FFB

- Call, 319, 327
- Change, FBD, 187
- Change, LD, 220
- Create, FBD, 186
- Create, LD, 219
- FBD, 178
- Insert, FBD, 186
- Insert, LD, 219
- Invocation, 321, 373, 377
- LD, 209
- Position, 186, 219
- Replace, FBD, 187
- Replace, LD, 220

Function

- FBD, 178
- LD, 209

Function Block

- FBD, 179
- LD, 210

Function Block language, 173

Function Blocks for Interrupt Sections, 1065

G

General, 1

- Backplane Expander, 99
- Hardware configuration, 71
- Loading a project, 598
- Online control panel, 581
- Online functions, 564
- OFFLINE and ONLINE mode, 75
- PLC configuration, 72
- PLC Connection, 566
- Reference Data Editor, 532
- Select process information, 593
- Variables editor, 480

Generate

- Project symbol, 743

Global data transfer

- Peer Cop, 867
- Global derived data type, 505

- Global DFB, 420
 - Defining the Path, 1033
 - INI File, 1033
 - Reading, 1034
 - Storing, 1034
- Global macro, 460
- Global Variables in DFBs, 430

H

- Hardware
 - Performance, 707
- Head setup, 53
- Help, 746
- Help Files
 - Defining the Path, 1033
- How to use the Ethernet / I/O Scanner
 - Ethernet / I/O Scanner, 109

I

- I/O Event Sections, 1060
 - Handling, 1041
 - Priority, 1061
 - Runtime Error, 1062
- I/O map, 52, 87
- Icon bar, 753, 754, 755, 756, 758
- Icons, 751, 753, 754, 755, 756, 758, 759, 760, 762
- Icons_Project Browser, 762
- Identifier, 262
- IEC
 - Hot Standby data, 83
 - Momentum first startup, 962, 999, 1017
- IEC conformity, 779
- IEC section
 - Animation, 607
- IL, 279
 - Animation, 334, 337
 - Block call up, 320
 - Code generation, 332
 - Creating a program, 339
 - Instruction, 283, 284
 - List of Symbols, 759
 - Modifier, 287
 - Online functions, 333, 334
 - Operands, 285
 - Operators, 288, 295
 - Short Cut Keys, 765
 - Syntax check, 330
 - Tag, 291
- IL command
 - call function block, 321
 - Compare, 310, 311, 314
 - Comments, 294
 - Compare, 312, 313, 315
 - Declaration, 292
 - DFB invocation, 321
 - Function call, 327
 - Invert, 305
 - Reset, 299
 - Set, 298
 - VAR...END_VAR, 292
- IL operation
 - addition, 306
 - Boolean AND, 300
 - Boolean exclusive OR, 304
 - Boolean OR, 302
 - Call DFB, 319
 - Call function block, 319
 - Division, 309
 - Jump to label, 316
 - Load, 296
 - Multiplication, 308
 - Store, 297
 - Subtraction, 307
- Import, 619
 - General Information, 622
 - INTERBUS configuration, 884
 - PLC Configuration, 658
 - Profibus DP configuration, 856
 - Section, 630, 635, 645, 646, 647
 - Structured variables, 653
 - Variables, 649, 653, 656
- INC
 - Include File, 507
- Include File
 - Extended Data Type Definition, 507

- INI File, 1027
 - CONCEPT.INI, 1029
 - Event Viewer Settings, 1039
 - General Information, 1030, 1039
 - LD section settings, 1035
 - Path for Global DFBs, 1033
 - Path for Help Files, 1033
 - Path for MBPPATH.INI, 1033
 - Print settings, 1031
 - Project Name Definition, 1032
 - Projectname.INI, 1038
 - Project Specific, 1027
 - Reading Global DFBs, 1033
 - Register Address Format Settings, 1032
 - Representation of internal data, 1035
 - Security Settings, 1037
 - Setting for Online Processing, 1036
 - Settings for the Address Format, 1036
 - Settings for Warning Messages, 1036
 - Storage of Global DFBs during Upload, 1033
 - Variable Storage Settings, 1032
 - Initial step, 238
 - Insert
 - FFB, FBD, 186
 - FFB, LD, 219
 - Install
 - Loadables, 52
 - EXEC file, 1024
 - Modbus Plus driver
 - Windows 98/2000/NT, 939
 - Install the SA85/PCI85
 - Modbus Plus Preferences, 934
 - Windows NT, 937
 - Windows 98/2000/XP, 934
 - Instruction
 - IL, 283, 284
 - ST, 358
 - Instruction list, 279
 - INTERBUS controller, 836
 - INTERBUS Controller with Atrium, 878
 - INTERBUS export settings in CMD, 879
 - Interface Settings in Windows 98/2000/XP
 - Modbus Preferences, 948
 - Interface settings in Windows NT
 - Modbus Preferences, 950
 - Interrupt Processing, 1041
 - General, 1044
 - Interrupt Sections
 - Disable, 42
 - EFBs, 1065
 - Examples for Setting Parameters, 1054
 - Execution Order, 1051
 - I/O Event Sections, 1060
 - Operating System, 1052
 - Priority, 1061
 - Runtime Error, 1062
 - Scan Rate for Timer Event Sections, 1048
 - Timer Event Sections, 1047, 1049
 - Invocation
 - DFB, 321, 373
 - FFB, 321, 373, 377
 - Project, 743
- J**
- Jump
 - SFC, 246
- K**
- Key combinations, 751, 763, 764, 765, 768, 771, 777
 - Key words
 - data type editor, 512
 - derived data type, 512
 - Keys, 751, 763, 764, 765, 768, 771, 777
- L**
- Ladder Diagram, 199
 - Ladder Logic 984, 387
 - LD, 199
 - Actual parameters, 215
 - Animation, 226
 - Calling a macro, 476
 - Closer, 205
 - Code generation, 224
 - Coil - negated, 207
 - Coil – negative edge, 208

-
- Coil – positive edge, 207
 - Coil - reset, 208
 - Coil - set, 208
 - Coils, 206
 - Contacts, 205, 206
 - Data flow, 221
 - Derived function block, 211
 - EFB, 209
 - Elementary function, 209
 - EN, 213
 - ENO, 213
 - Execution sequence, 221
 - FFB, 209
 - Function, 209
 - Function block, 210
 - Icon bar, 758
 - Link, 214
 - Loops, 221
 - Online functions, 226
 - Opener, 205
 - Program creation, 229
 - Shortcut keys, 771
 - Text object, 217
 - UDEFB, 212
 - User-defined elementary function, 212
 - Learn monitoring times
 - SFC, 274
 - Libraries, 8
 - Limitations
 - LL984, 391
 - Link
 - FBD, 182
 - LD, 214
 - List of Symbols, 751, 759, 760
 - List of Tools, 751, 759, 760
 - Literals, 35
 - LL984, 387
 - Close Column, 398
 - Combination mode, 414
 - Dialog interaction, 394
 - Direct programming, 414
 - DX Zoom, 400
 - Edit, 393, 397
 - Editing Networks, 398
 - Equation network, 404, 405
 - Equation network, Syntax and Semantics, 409
 - List of Symbols, 760
 - Momentum first startup, 967, 983, 1003, 1020
 - Navigation, 393
 - Online Restriction, 394
 - Online Search, 401
 - Open Column, 398
 - Open Row, 398
 - Programming modes, 413, 414
 - Reference Offset, 396
 - Reference Zoom, 399
 - References, 395
 - Replace References, 401
 - Requirements, 393
 - Section, 390
 - Segment, 390
 - Select, 397
 - Short Cut Keys, 777
 - Subroutines, 402
 - Trace, 401
 - Undo, 397
 - Variables, 395
 - LL984 Processing
 - speed optimized, 585
 - LL984 section
 - Animation, 609
 - Load reference data, 542
 - Loadables, 84
 - Atrium, 166
 - compact, 150
 - CPU 424 02, 131
 - CPU X13 0X, 131
 - CPU 434 12, 139
 - CPU 534 14, 139
 - Loading, 599
 - Loading a project, 597
 - General information, 598
 - Loading firmware, 1024
 - Local derived data type, 505
 - Local DFB, 420
 - Local macro, 460
 - Located variables
 - Changing signal states in RDE, 535
 - Log Encrypting, 15
 - LOG File, 613, 614

Logging

LOG File, 614

Logging Write Access to the PLC, 613

Loop

FBD, 189

LD, 221

M

Macro, 455, 458

Calls from SFC, 473

Calls from FBD, 476

Calls from LD, 476

Context sensitive help, 465

Convert, 911

Create, 467

Delete, 676

Documentation, 663

Exchange marking, 462

Global, 460

Local, 460

Maximum supervision time, 238

MBPPATH.INI

Defining the Path, 1033

MBX Driver

Driver for connection between

ModConnect Host interface adapters and

32 bit applications with Windows 98/

2000/NT, 941

Memory, 115

Optimize, 119

Structure, 117

Memory and optimization

Atrium, 163

Compact, 147

Momentum, 157

Quantum, 122, 136

Memory partitions, 51

Memory statistics, 595

Menu commands, 737

Minimum configuration, 51

Minimum supervision time, 239

MMS-Ethernet

Specify coupling modules, 92

Modbus

Compact first startup, 958, 977

Momentum first startup, 962, 967, 980, 983

Quantum first startup, 954, 974

Startup with DOS Loader, 973

Startup with the EXECLoader, 953

Modbus communication, 53

Modbus network link, 570

Modbus Plus

Atrium first startup, 996, 1014

Compact first startup, 992, 1011

Momentum first startup, 999, 1003, 1017, 1020

Quantum first startup, 988, 1008

Remote MBX Driver, 942

Startup with DOS Loader, 1007

Startup with the EXECLoader, 987

Virtual MBX Driver, 940

Write Restriction, 112

Modbus Plus Bridge, 576

Modbus Plus Network Connection, 571

Modbus Plus network node, 93

Modbus Plus Preferences

Installing the SA85/PCI85, 934

Installing the Modbus Plus driver in

Windows 98/2000/NT, 939

Establishing the hardware connection, 945

Startup, 933

Modbus Plus Routing Path

Automatic Connection, 1068, 1071

Modbus Preferences

Interface Settings in Windows 98/2000/XP, 948

Interface Settings in Windows NT, 950

Transfer problems, 951

Establishing the hardware connection, 950

Startup, 947

ModConnect, 915

MODIFIED, 566

Modifier

IL, 287

Modsoft
 Conversion, 923
 Function compatibility, 932
 References, 929
 Momentum
 Memory optimization, 157
 Momentum example
 Ethernet bus system, 895
 Remote I/O bus, 887
 Momentum first startup
 DOS Loader, 980, 983, 1017, 1020
 EXECLoader, 962, 967, 999, 1003
 Modbus, 962, 967, 980, 983
 Modbus Plus, 999, 1003, 1017, 1020
 MSTR-Read-Operation, 113

N

Names
 Datatype editor, 516
 Derived datatype, 516
 Navigation
 LL984, 393
 Network Configuration
 TCP/IP, 897
 Network Connection
 Modbus Plus, 571
 Network link
 Modbus, 570
 TCP/IP, 578
 NOM/NOE
 Disable Write Access, 112
 NOT EQUAL, 566

O

Objects
 Insert, LD, 219
 SFC, 237
 Offline functions in the configurator, 76
 Online, 679, 682
 INI File, 1036
 SFC, 269
 Online Control Panel, 581, 585, 589
 Online diagnosis, 610

Online functionen, 14, 561
 Configurator, 76
 General information, 563, 564
 FBD, 193
 IL, 333
 IL/ST, 334
 LD, 226
 ST, 382
 SFC, 270, 272
 Online help, 746
 ONLINE Operation
 Presettings, 569
 Online Restriction
 LL984, 394
 Online Search
 LL984, 401
 Open
 Project, 743, 744
 Open Column
 LL984, 398
 Open Row
 LL984, 398
 Opener
 LD, 205
 Operands
 IL, 285
 ST, 346
 Operating System
 Timer Event Sections, 1052
 Operators
 IL, 288, 295
 ST, 347
 Optimize
 PLC Memory, 119
 Optional Configuration, 90

P

Page breaks for sections, 667
 Parallel branch, 251
 Parallel connection, 252
 Parameterize ASCII interface, 94
 Parameterize interfaces
 ASCII interface, 94
 Modbus interface, 94
 Parameterize Modbus interface, 94

- Parameters for Automatic Connection, 744
- Password protection, 691
- Path for Global DFBs
 - Settings in the INI File, 1033
- Path for Help Files
 - Settings in the INI File, 1033
- Peer Cop, 93, 864
- Peer Cop communication, 54
- Performance
 - hardware, 707
 - PLC family, 707
- Phase
 - Timer Event Sections, 1049
- PLC
 - Simulating, 677
 - Status, 733
- PLC configuration, 50, 51, 69
 - Export, 658
 - General information, 72
 - Icons, 761
 - Import, 658
- PLC Connection
 - General, 566
- PLC family
 - Performance, 707
- PLC Memory, 115
 - Optimize, 119
 - Structure, 117
- PLC Memory and optimization
 - Atrium, 163
 - Compact, 147
 - Momentum, 157
 - Quantum, 122, 136
- PLC memory mapping, 83
- PLC selection, 80
- PLC State, 566, 579, 594
- Position
 - FFB, FBD, 186
 - FFB, LD, 219
- Precondition for unconditional configuration, 79
- Presettings for Modbus
 - Startup, 947
- Presettings for Modbus Plus
 - Startup, 933
- Presettings for ONLINE operation, 569
- Print
 - Settings in the INI file, 1031
 - Sections, 667
- Priority
 - I/O Event Sections, 1061
- Proceed in the following way with the configuration, 73
- Process
 - Actions, 259
 - Program, 30
 - Project, 30
 - Step properties, 257
 - Transition, 264
- PROFIBUS
 - Specify coupling modules, 92
- Profibus DP controller, 850
- Profibus DP export settings in SyCon, 850
- Program
 - Create, 47
 - Processing, 30
 - Status, 733
 - Structure, 29, 30
- Program creation
 - FBD, 196
 - LD, 229
 - ST, 385
- Programming, 6
- Programming languages, 9
- Programming modes
 - LL984, 413, 414
- Programs, 35
- Project
 - Archiving, 674
 - Call, 744
 - Convert, 911
 - Create, 47
 - Delete, 676
 - Documentation, 663
 - Invoke, 743
 - Open, 743, 744
 - Processing, 30
 - Protect, 702
 - Structure, 29, 30

- Project Browser, 491
 - Keyboard operation, 496
 - Mouse operation, 496
 - Toolbar, 762
- Project Name Definition
 - INI File Settings, 1032
- Project Symbol
 - Generate, 743
 - Create, 744
- Projectname.INI, 1027, 1038
 - Event Viewer Settings, 1039
 - General Information, 1039
- Protect
 - DFB, 702
 - Project, 702

Q

- Quantum
 - Memory optimization, 122, 136
- Quantum example
 - INTERBUS control, 835
 - Profibus DP controller, 849
 - Quantum-Peer Cop, 863
 - Remote control with DIO, 826
 - Remote control with RIO, 807
 - Remote control with RIO (series 800), 815
 - SY/MAX controller, 841
- Quantum first startup
 - DOS Loader, 974, 1008
 - EXECLoader, 954, 988
 - Modbus, 954, 974
 - Modbus Plus, 988, 1008
- Quantum Security Parameters, 112

R

- Range Monitoring
 - ARRAY, 527
- RDE, 531
 - Converting RDE templates, 533
 - Cyclical Setting of Variables, 536
 - General, 532
 - Toolbar, 762
- Reactivate flash save, 588

- Reading Global DFBs
 - Settings in the INI File, 1033
- Reference data editor, 531
 - Changing signal states of a Located variable, 535
 - Converting RDE templates, 533
 - Cyclical Setting of Variables, 536
 - General, 532
 - Replacing variable names, 541
- Reference Offset
 - LL984, 396
- Reference Zoom
 - LL984, 399
- References
 - LL984, 395
- Register Address Format
 - INI File Settings, 1032
- Remote controller with DIO, 831
- Remote controller with RIO, 812
- Remote controller with RIO (series 800), 820
- Remote MBX Driver
 - Modbus Plus, 942
- Replace
 - coil, LD, 220
 - contact, LD, 220
 - FFB, FBD, 187
 - FFB, LD, 220
 - Variable names, 541
- Replace References
 - LL984, 401
- Requirements
 - LL984, 393
- RTU extension
 - Compact configuration, 105
 - Configure, 105
- Runtime Error
 - I/O Event Sections, 1062

S

- Save To Flash, 585
- Scan
 - Constant, 582
- Scan rate
 - Timer Event Sections, 1048

- Scan times
 - single, 583
- Search and Replace
 - Variable names and addresses, 483
- Search and paste
 - Variable names and addresses, 487
- Section, 40
 - Animation, 606
 - Disable, 42
 - Execution order, 41
 - Export, 625
 - Import, 630, 635, 645, 646, 647
 - Import, 631, 642
 - LL984, 390
 - Status, 733
- Secure Application, 15
- Security, 691, 692, 694, 701, 702
- Segment
 - LL984, 390
- Segment manager, 86
- Select
 - LL984, 397
- Select process information
 - General information, 593
 - Status and memory, 592
- Selecting process information
 - Status and memory, 592
- Separators
 - Data type editor, 517
 - Derived data type, 517
- Set/Change PLC Password, 589
- Setting up and controlling the PLC, 580
- Setup and control PLC
 - General information, 581
- SFC
 - 'SFCSTEP_STATE' variable, 240
 - 'SFCSTEP_TIMES' variable, 239
 - Action, 240, 259
 - Action variable, 241
 - Alternative branch, 248
 - Alternative connection, 250
 - Animation, 270, 272
 - Calling up macros, 473
 - Edit, 253
 - Icon bar, 756
 - Identifier, 262
 - Initial step, 238
 - Jump, 246
 - Learn monitoring times, 274
 - Link, 245
 - Maximum supervision time, 238
 - Minimum supervision time, 239
 - Objects, 237
 - Online, 269
 - Online functions, 270, 272
 - Parallel branch, 251
 - Parallel connection, 252
 - Short Cut Keys, 768
 - Step, 238
 - Step delay time, 238
 - Step duration, 238
 - Step properties, 257
 - String, 272
 - Text object, 252
 - Transition, 242, 264
 - Transition diagnosis, 277
 - Transition section, 243
 - Transition variable, 244
 - Waiting step, 238
- Short cut keys, 751, 763
- Simple sequences, 245
- Simulation, 677, 679, 682
 - SPS, 679, 682
- Single sweeps, 583
- Special options, 96
- Specific data transfer
 - Peer Cop, 869
- Speed optimized LL984- Processing, 585
- SPS
 - Simulate, 679, 682
- ST, 341
 - Animation, 334
 - Assign instructions, 357
 - Block call up, 372
 - Code generation, 381
 - Expressions, 345
 - Instructions, 358
 - List of Symbols, 759
 - Online functions, 334, 382
 - Operands, 346
 - Operators, 350
 - operators, 347

-
- Program creation, 385
 - Short Cut Keys, 765
 - syntax check, 380
 - ST Command
 - , 352, 354
 - (), 351
 - *, 352
 - **, 351
 - +, 353
 - >, 354
 - >=, 354
 - Addition, 353
 - , 355, 355
 - &, 356
 - =, 354
 - /, 353
 - AND, 356
 - Assignment, 359
 - Boolean AND, 356
 - Boolean Exclusive OR, 356
 - Boolean OR, 356
 - Call function block, 373
 - CASE...OF...END_CASE, 365
 - Complement formation, 352
 - Declaration, 360
 - Division, 353
 - ELSE, 363
 - ELSIF...THEN, 364
 - Empty instruction, 371
 - Equal to, 354
 - EXIT, 371
 - Exponentiation, 351
 - FOR...TO...BY...DO...END_FOR, 366
 - FUNCNAME, 351
 - function invocation, 377
 - Greater than, 354
 - Greater than/Equal to, 354
 - IF...THEN...END_IF, 362
 - Less than, 355
 - Less than or equal to, 355
 - MOD, 353
 - Modulo, 353
 - Multiplication, 352
 - Negation, 352
 - NOT, 352
 - Not equal to, 355
 - OR, 356
 - REPEAT...UNTIL...END_REPEAT, 370
 - Subtraction, 354
 - Use of parentheses, 351
 - VAR...END_VAR, 360
 - WHILE...DO...END_WHILE, 368
 - XOR, 356
 - ST Comment
 - Comment, 371
 - Start behavior
 - Variable, 37
 - Digital outputs, 39
 - Startup
 - Presettings for Modbus, 947
 - Presettings for Modbus Plus, 933
 - Startup with DOS Loader
 - Modbus, 973
 - Modbus Plus, 1007
 - Startup with the EXECLoader
 - Modbus, 953
 - Modbus Plus, 987
 - State of the PLC, 579
 - Status, 566
 - Status bar, 733
 - Step, 238
 - Alias designations, 266
 - Step delay time, 238
 - Step duration, 238
 - Step properties
 - Process, 257
 - Storage of Global DFBs during Upload
 - Settings in the INI File, 1033
 - String
 - Control, 272
 - Structure
 - PLC Memory, 117
 - Program, 29, 30
 - Project, 29, 30
 - Structured text, 341
 - Structured variables
 - Import, 653
 - Subroutines
 - LL984, 402
 - Symax-Ethernet
 - specify coupling modules, 92
 - Symbols, 751, 759, 760
-

Syntax

- Data type editor, 509
- Derived Data Type, 509

Syntax check

- IL, 330
- ST, 380

T

Tag

- IL, 291

TCP/IP

- Network Configuration, 897
- Network link, 578

TCP/IP-Ethernet

- specify coupling modules, 92

Text Object

- FBD, 184
- LD, 217
- SFC, 252

Timer Event Sections, 1047

- Define Scan Rate, 1048
- Defining the Phase, 1049
- Examples for Parameterization, 1054
- Execution Order, 1051
- Handling, 1041
- Operating System, 1052

Toolbar, 753, 754, 755, 756, 758

Tools, 17

Trace

- LL984, 401

Transfer problems

- Modbus Presettings, 951

Transition, 242

- Alias designations, 266
- Declare, 264
- Process, 264

Transition diagnosis, 277

Transition section, 243

Transition variable, 244

U

UDEFB

- FBD, 181
- LD, 212

Unconditional Configuration, 78

- Precondition, 79

Unconditional locking of a section, 538

Undo

- LL984, 397

Uploading PLC, 603

User-defined Elementary Function

- FBD, 181
- LD, 212

Utility program, 17

V

Variable Editor, 479

- Declaration, 480
- Exporting located variables, 490
- Search and replace, 483
- Search and paste, 487

Variable Storage

- INI File Settings, 1032

Variables, 35

- ASCII message editor, 548
- Export, 629
- Import, 649, 653, 656
- LL984, 395
- Start behavior, 37

VARINOUT variables, 423

Various PLC settings, 56

View Tool, 614

Virtual MBX Driver

- Modbus Plus, 940

W

Waiting step, 238

Warm restart, 37

Window elements, 733

Window types, 732

- Windows, 729
 - Check box, 742
 - Command buttons, 741
 - Dialog boxes, 740
 - Lists, 741
 - Menu commands, 737
 - Option buttons, 741
 - Status bar, 733
 - Text boxes, 741
 - Window, 731
 - Window elements, 733
 - Window types, 732

